INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

NOTE TO USERS

The original manuscript received by UMI contains pages with indistinct, light, broken, and/or slanted print. Pages were microfilmed as received.

This reproduction is the best copy available

UMI

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A

Volume I

Distributed Intelligent Information Network

by

Apurba K Saha

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

1999

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

UMI Number: 9924846

Copyright 1999 by
Saha, Apurba K.

All rights reserved.

UMI Microform 9924846
Copyright 1999, by UMI Company. All rights reserved.

This microform edition is protected against unauthorized
copying under Title 17, United States Code.

UMI
300 North Zeeb Road
Ann Arbor, MI 48103

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Copyright

1999

Apurba K Saha

All Rights Reserved

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Date	Chair of Examining Committee
01/27/99	Executive Officer The City University Of New York
Date	Professor Michael Anshell The City University Of New York
	Professor Charles Giardina The City University Of New York
	Dr. Bharat Doshi (Lucent Technologies)
	Dr. Victor B. Lawrence (Lucent Technologi
	Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Abstract

Distributed Intelligent Information Network

By: Apurba K Saha

Advisor: Professor Syed V. Ahamed

A global network-based information model called Distributed Intelligent Information Network (DIIN) has been designed for online information retrieval using existing education databases over the public and intelligent network. Such a globally networked application system can be used by researchers in university campuses from all over the world and will include considerable number of new features. This model is based upon object-oriented paradigm using object-oriented model and a negotiating protocol of distributed Artificial Intelligence. This model has been described and analyzed in great details.

On the way of presenting this, Chapter 1 provides an introduction to Information Retrieval (IR), Chapter 2 provides a thorough accounting of different topics of IR, Chapter 3 presents a detailed account of high-speed networks and cellular networks. Chapter 4 describes Intelligent networks, knowledge representations and Distributed Artificial Intelligence (DAI). It also discusses the architecture of a conceptual network (called NBIS) for information retrieval. Chapter 5 describes the proposed network architecture (DIIN-IR), its software components, message protocols etc. in full details. Chapter 6 contains an appendix.

This work is being submitted as the Ph.D. dissertation thesis for the author.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Acknowledgment

In the beginning, I acknowledge my gratitude to my advisor Prof Syed V. Ahamed of CUNY for his inspiring guidance, valuable help and thoughtful suggestions to make this project a success. His sincere help and positive cooperation has been the major force to drive me on without which this work would not have completed.

At the same time, I really appreciate the help, encouragement and full cooperation which I received from Dr. Victor B. Lawrence, the director of Advanced Communications Laboratory of Lucent Technologies, a leading thinker as well as a profound researcher in technologies of modern communications. His remarks were significantly helpful in my achieving success in the project.

Among many cooperative professors of CUNY, Prof Stanley Habib has been a constant source of cooperation and guidance throughout the project to whom I am absolutely indebted. I truly acknowledge my gratitude to Prof Michael Kress, Prof Charles Giardina, Prof. Rohit Parikh, Prof. Michael Anshel, Mr. Joseph Driscoll and staff members of the Computer Center at GSUC, CUNY for offering their kind assistance and advice during the course of this research. Without their support this project could not even be started.

In addition, I am grateful to Mr. Kevin Vroman, Lisa Revel, David Hoibak and other staff members from DMR-TRECOM for sponsoring this project. I am also grateful to Mr. Edward Fudal, Joseph Devine, Jeffrey Portland from AT&T whose technical support, flexible attitude and sympathy let this project continue.

Finally, I am thankful to my wife Sipra Saha and all my family members for their sacrifice that they bore and the encouragement they offered to this work.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table of Contents (Volume I)

1. OVERVIEW OF INFORMATION RETRIEVAL 2

1.1 INTRODUCTION. 2
1.2 EDUCATIONAL INFORMATION RETRIEVAL 2
1.3 FEATURES OF EXPERT SEARCHER 3
1.4 OVERVIEW OF IR 4
1.5 FUNDAMENTAL ISSUES 4
1.6 DEFINITIONS 5
1.6.1 Conceptual Model 7
1.6.2 Data Structures & File Structures 10
1.6.3 Query Operations 13
1.6.4 TERM OPERATIONS 14
1.6.5 Document Operations 16
1.6.6 Relevance Feedback for term 16
1.6.7 Document Correlation 16
1.6.8 TERM-WEIGHTING 17
1.6.9 Scope Adjustment 20
1.6.10 Semantic Modeling 21
1.6.11 Linguistic Model 22
1.6.12 Object-Oriented Model 23
1.6.13 Syntagmatic model 24

2. AUTOMATED INFORMATION INDEXING AND RETRIEVAL 28

2.1 INTRODUCTION 28
2.2 LANGUAGE ANALYSIS 28
2.3 DICTIONARY CONSTRUCTION 30
2.4 CONCEPT HIERARCHY DICTIONARY 34
2.5 DICTIONARY PERFORMANCE 35
2.5.1 The Stem Thesaurus 35
2.5.2 The regular thesaurus 36
2.5.3 The phrase dictionary 36
2.6 AUTOMATIC THESAURUS CONSTRUCTION 37
2.6.1 Fully automatic method 37
2.6.2 Semiautomatic methods 39
2.7 AUTOMATIC HIERARCHY ARRANGEMENT 40
2.8 INDEX STRUCTURE REPRESENTATION 43
2.9 STATISTICAL PHRASE PROCESSING 44
2.10 STATISTICAL SIMILARITY OPERATIONS 45
2.11 IMPLEMENTATION SCHEME OF ASSOCIATIVE RETRIEVAL 50
2.12 EXPERIMENTAL RESULTS FOR CONCEPT CORRELATION 55
2.13 DOCUMENT CLUSTERING 56
2.13.1 eigenvalue and factor analysis 57
2.14 OVERLAPPING CLUSTERING 61
2.15 TERM-SENTENCE CLUSTERING 67
2.16 SYNTACTIC OPERATIONS 69
2.16.1 Automatic content analysis 69
2.16.2 Syntax and Semantics 71
2.16.3 Phrase structure parsing 73
2.16.4 Transformational Grammar 75
2.16.5 Semantics 77

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.17 CRITERION PHRASES 78
2.17.1 Criterion phrase format 80
2.17.2 Syntactic phrase processing 85
2.17.3 Syntactic tree matching 88
2.17.4 Quality 105
2.18 ALGEBRAIC ABSTRACT MODEL FOR RETRIEVAL 106
2.18.1 Inclusive information retrieval 107
2.19 CORRELATION FORMULAE FOR DOCUMENT VECTORS 127
2.20 FILE STRUCTURES 131
2.21 ITERATIVE SEARCH USING RELEVANCE FEEDBACK 141
2.22 ASSESSMENT OF INFORMATION RETRIEVAL SYSTEMS 145
2.23 METHODS FOR DETERMINING THE RECALL VALUE 155
2.24 PRESENTATION OF RESULTS 156
2.25 ANALYSIS OF AUTOMATED INFORMATION SYSTEMS 172
2.25.1 Test environment 173
2.26 ADDITIONAL INFORMATION SERVICES 192
2.26.1 Text Transformation 192
2.26.2 Index generation 199

3. HIGH-SPEED NETWORKING & CELLULAR NETWORKS 200

3.1 NETWORKS 201
3.2 INTRODUCTION TO HIGH-SPEED NETWORKING 202
3.3 OPTICAL FIBER 205
3.4 ASYNCHRONOUS TRANSFER MODE 210
3.4.1 Adaptation Layer protocols 212
3.4.2 ATM Cell Format 214
3.4.3 Design of Adaptation Layer 217
3.5 MULTIMEDIA APPLICATIONS 224
3.6 BOTTLENECK IN SOFTWARE FOR HIGH-SPEED NETWORKING 228
3.6.1 The Afterburner Interface 229
3.6.2 Copy-on-Write with DMA 231
3.6.3 Other Processor-memory Interactions 233
3.6.4 Architecture of TCP/IP 233
3.7 PERFORMANCE GUARANTEES 244
3.7.1 Statistical Multiplexing 245
3.7.2 Weighted Fair Queuing 247
3.7.3 Jitter Control scheme 250
3.7.4 Flow Setup and Routing in Internetwork 255
3.7.5 ATM Connection Setup 268
3.7.6 Routing 271
3.8 CELLULAR AND CORDLESS COMMUNICATIONS 274
3.8.1 Cellular network architecture 275
3.8.2 Cordless Network architecture 278
3.8.3 Third generation Architecture 278
3.8.4 Transmission Technologies 281
3.9 MOBILE IP 292
3.9.1 Protocol Overview 295
3.10 INTERNET AND ITS ARCHITECTURE 310
3.10.1 National Internet Backbone for USA 311
3.11 IPv6 317
3.11.1 Addressing and Routing strategy in IPv6 317
3.11.2 Autoconfiguration scheme 318
3.11.3 More streamlined format 319
3.11.4 Data security 319

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.11.5 Multicast and anycast 320
3.11.6 Anycast Services 321
3.11.7 Quality of Service 322
3.11.8 Transition to IPv6 323

4. DISTRIBUTED INTELLIGENT INFORMATION NETWORK 327

4.1 INTELLIGENT NETWORKS(IN) 327
4.1.1 Evolution of IN 328
4.1.2 IN architecture 329
4.1.3 International Standards 332
4.1.4 Conceptual view of IN 333
4.2 ARTIFICIAL INTELLIGENCE & KNOWLEDGE-BASE 336
4.2.1 Introduction 336
4.2.2 Knowledge-Base Systems 337
4.2.3 Rule based expert system 354
4.2.4 Slot and Filler Knowledge Representation 378
4.3 INFORMATION NETWORK 405
4.3.1 Basic Concept 406
4.3.2 Objectives Of INN 406
4.3.3 Evolution of IN to INN 408
4.3.4 Fabric of INN 408
4.3.5 Major elements of INN 409
4.4 DISTRIBUTED ARTIFICIAL INTELLIGENCE 413
4.4.1 Introduction 413
4.4.2 Principles of Action 413
4.5 A NETWORK-BASED INFORMATION SYSTEM (NBIS) 416
4.5.1 Introduction 416
4.5.2 Concept behind NBIS 417
4.5.3 Role of knowledge-base 417
4.5.4 Description of NBIS 418
4.5.5 NBIS network protocol 422
4.5.6 Architecture of NBIS 423
4.5.7 Intelligent functions of NBIS 423

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table of Contents (Volume II)

5. DISTRIBUTED INTELLIGENT INFORMATION NETWORK-INFORMATION RETRIEVAL (DIIN-IR) 425

5.1 INTRODUCTION 425
5.2 BASIC PRINCIPLE 425
5.3 SOFTWARE ARCHITECTURE 427
5.3.1 Linguistic Processing layer (LL) 427
5.3.2 Generalization Layer 429
5.3.3 Search Initiator Layer (SIL) 431
5.4 QUERY RELAXATION 433
5.4.1 Formal Definition 434
5.4.2 User constraints 437
5.4.3 Limiting Relaxation with User Constraints 444
5.4.4 Heuristics for Ordering Relaxations 447
5.4.5 Processing Queries 452
5.4.6 Comments 453
5.5 DISTRIBUTED KNOWLEDGE-BASE REPRESENTATION USING CSP 454
5.5.1 Background 455
5.5.2 Alterations to the original algorithm 456
5.5.3 The updating Algorithm 458
5.5.4 Examples 462
5.6 ARTIFICIAL INTELLIGENCE FOR COOPERATIVE INFORMATION SYSTEMS 466
5.6.1 Cooperative Information systems 466
5.6.2 Knowledge architecture in Cooperative Information systems 478
5.6.3 Modeling_Unit Structure 480
5.6.4 Modeling_Unit Behavior 482
5.6.5 Translations Rules into Colored Nets 485
5.6.6 Modeling_Unit Validation 486
5.6.7 Design of Intelligent Cooperative Societies 487
5.6.8 Knowledge Base for Representation and analysis of Petri Nets 493
5.6.9 Modular Implementation issues 494
5.6.10 Introduction to Inter-schema knowledge-based system 496
5.7 COMPONENTS, ALGORITHMS & PROCEDURES 507
5.7.1 Overview 507
5.7.2 Functional components 508

6. APPENDIX A 531

Bibliography ..... 6[illegible]

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of figures

Figure 1:Variation of Recall ratio with precision of queries 6
Figure 2:Taxonomy of conceptual model of IR 7
Figure 3:Simplistic Thesaurus with concept number 11
Figure 4:Typical suffix table 15
Figure 5:Various operations on terms 15
Figure 6:deep structure on phrase-structure grammar 22
Figure 7:Object classification 23
Figure 8:Taxonomy of today's research areas in IR 26
Figure 9:Thesaurus excerpt in concept no order 30
Figure 10:Thesaurus excerpt in alphabetical order 30
Figure 11:Excerpt of a syntactic tree 32
Figure 12:Criterion phrase description 33
Figure 13:Concept hierarchy 35
Figure 14:Performance for unweighted indexing 36
Figure 15:Performance for weighted indexing 36
Figure 16:Comparison based on phrase dictionary 37
Figure 17:Term-Document Matrix 38
Figure 18:term-Document Graph 38
Figure 19: Term groupings for Dictionary 39
Figure 20:Hierarchy arrangement 40
Figure 21: term-term similarity matrix Reduced similarity matrix 42
Figure 22:A word tree for word-stems Add, Addict 44
Figure 23:converting an m-ary tree to a binary tree 44
Figure 24:A typical entry in chained concept list 45
Figure 25:Example document-term matrix 51
Figure 26:Chained document-concept matrix 52
Figure 27:Example concept-concept correlation process 53
Figure 28:Flowchart for concept association 54
Figure 29:Frequency characteristics of concept-concept correlation 56
Figure 30:reducible symmetric matrix (b) corresponding directed graph 58
Figure 31:Failure of density test 63
Figure 32:Depiction of Rocchio's clustering process 65
Figure 33:Simple clustering method 66
Figure 34:An example concept-sentence matrix 68
Figure 35:Simple semantic graph for phrase showing synthetic relation 71
Figure 36:Sentence derivation 72
Figure 37:Converting a phrase marker into a dependency tree 72
Figure 38:Semantics and syntax for a given sentence 76
Figure 39: A modified dependency tree 79
Figure 40: Embedded phrases 81
Figure 41:Sample syntactic phrase specification 82
Figure 42:Syntactic phrase and tree type formats 83
Figure 43:Internal storage format for syntactic phrase INFRET 85
Figure 44:Flow-diagram for syntactic phrase processing 85
Figure 45:Creation of sentence tree from syntactic structure code 87
Figure 46:directed graphs 89
Figure 47:Connection matrix for graphs in fig 46 89
Figure 48: Flow-diagram for graph-matching procedure 94
Figure 49:Topological matching of partial substructure 94
Figure 50:Path matrices (computed from the connection matrices) 95
Figure 51:Directed graphs including branch labels 96

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Figure 52:Branch label matrices 96
Figure 53:Syntol document graph and Syntol query graph 97
Figure 54:Typical syntactic phrase 99
Figure 55:Typical document sentence syntactic tree 100
Figure 56:{(a),(b),(c),(d)} Tree structures matching the syntactic phrase of fig 54. 101
Figure 57:Example criterion phrase tree for phrase INFRET 102
Figure 58:Example for node-by-node tree matching 104
Figure 59:Request-Document mapping 107
Figure 60:Lattice elements as subsets of the set (a1, a2, a3) 108
Figure 61:Operations in Query space 110
Figure 62:Request-document mapping for combined requests 111
Figure 63:Description set C and query space R. 112
Figure 64:Retrieval model using document descriptions 113
Figure 65:Example retrieval system 115
Figure 66:Example retrieval model - one-to-one mapping 117
Figure 67:Complementation of collections 121
Figure 68:File structures for record-oriented storage 125
Figure 69:Key-oriented file structures 126
Figure 70:Inverted file organization using B-tree 132
Figure 71:Building inverted file including key directory 133
Figure 72:Pages of the inverted file with 2 overflow pages (combined file system) 134
Figure 73:Reference relations in a combined file system 135
Figure 74:Flow-chart for building a multi-list with key directory 136
Figure 75:A close picture of a multi-list (node K1 participates in 3 lists) 137
Figure 76:Odd queries in a clustered document space 140
Figure 77:Flow-chart for a simple relevance feedback system 142
Figure 78:Graphical illustration of relevance feedback 145
Figure 79:Four categories of documents 146
Figure 80:Graphical illustration of recall measure 148
Figure 81:Construction of normalized precision measure (n=5) 150
Figure 82:Normalized recall, precision and overall measures 162
Figure 83:Normalized recall, precision and overall measures for various methods 163
Figure 84:Typical automatic extraction system 198
Figure 85:Taxonomy of networks classifications 201
Figure 86:Protocol layer architectures 204
Figure 87:Cross-sectional view of a fiber 205
Figure 88:Beam propagation in multimode fiber 206
Figure 89: Logical view of SONET frames 208
Figure 90:SONET transport overhead bytes 209
Figure 91:B-ISDN protocol model for ATM 210
Figure 92:Hop routing in Cell networks 212
Figure 93:ATM Header at Network-Network interface (NNI) 214
Figure 94:Payload field format for user data cells 215
Figure 95:ATM header at User-Network interface (UNI) 216
Figure 96:AAL 1 Block for video 218
Figure 97:Breakdown of the adaptation layer 219
Figure 98:AAL 3 /4 SAR format 219
Figure 99: AAL 3 /4 Convergence Format 220
Figure 100:AAL 5 Convergence format 222
Figure 101:Bandwidth of video frames needing video encoding 227
Figure 102:The Afterburner interface layout 230
Figure 103:Afterburner FIFO Control Word Format 231
Figure 104:Architecture of TCP/IP protocol suite 235
Figure 105:A mini Internet configuration 235

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Figure 106: TCP fields in the Header 238
Figure 107:Model of leaky bucket 242
Figure 108:Token Bucket model 243
Figure 109:Stop-and-Go Queuing 250
Figure 110:RFC 1363 Flow Spec structure 257
Figure 111:Sequence of flow setup between the hops (ST-II) 263
Figure 112:RSVP delivery tree from Src to Rcv1, Rcv4 266
Figure 113:Adding a reservation from Rcv2 using RSVP 267
Figure 114:Q.93B Connection Setup 269
Figure 115:A Simple network (of routers and hosts) with link capacities 272
Figure 116:Simple network with constraints 272
Figure 117:Cellular Network Architecture 276
Figure 118:GSM Cellular network configuration 277
Figure 119:Future public land mobile telecommunications system model 279
Figure 120:Cellular packet switch 280
Figure 121:GSM multi-frame and frame structure 282
Figure 122:GSM time slot structure 282
Figure 123:IS-54 slots and frames 284
Figure 124:CT2 slot and frame 286
Figure 125:DECT slot and frame 287
Figure 126:Mobile IP datagram flow 296
Figure 127:Mobility Agent extension format 299
Figure 128:Data structure of a registration request 300
Figure 129:Registration Request Format 301
Figure 130:Registration reply format 302
Figure 131:Mobile IP authentication extensions 302
Figure 132:IP-within-IP encapsulation 305
Figure 133:Minimal IP encapsulation format 306
Figure 134:Various home network configurations 310
Figure 135:vBNS SCC architecture 313
Figure 136: vBNS backbone topology 314
Figure 137:vBNS PVPs over MCI's commercial ATM network 315
Figure 138:IPv6 header layout 319
Figure 139:Multicasting to a few subscribers 321
Figure 140:An example of Anycast 322
Figure 141:IPv6 over IPv4 Tunneling 324
Figure 142: A typical topology of two networks using an intelligent network 327
Figure 143:Architecture for AIN release 1.0 331
Figure 144:IN conceptual model 334
Figure 145:Levels of abstract in knowledge analysis 337
Figure 146:Various representations of the same knowledge 338
Figure 147:Search space tree 359
Figure 148:Search tree (goals) for goal-driven approach 362
Figure 149:A depth-first search 365
Figure 150:A breadth-first search 366
Figure 151:An implication network 373
Figure 152:Semantic net for a statement 'John punched his manager' 379
Figure 153:Examples of frames 381
Figure 154:Examples of frame instantiation 381
Figure 155:A frame-hierarchy structure 383
Figure 156:Example relationships between entity sets 386
Figure 157:Some KRL units 392
Figure 158:A simple restaurant script 404
Figure 159:Platform model for Information Network 406

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Figure 160:Abstraction of INN objects 410
Figure 161:Architecture layers for information network 410
Figure 162:Service interaction architecture 412
Figure 163:Arrangement of a knowledge-ring 419
Figure 164:A hierarchical bus arrangement of KSs 420
Figure 165:NBIS transaction flow diagram 422
Figure 166:Architecture of NBIS network 422
Figure 167:Layers of Software components 427
Figure 168:Subject-concepts matrix (S) 429
Figure 169:Subject-node dictionary 431
Figure 170:Internal organization of a blackboard 432

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of figures (Volume II)

Figure 167:Layers of Software components 427
Figure 168:Subject-concepts matrix (S) 429
Figure 169:Subject-node dictionary 431
Figure 170:Internal organization of a blackboard 432
Figure 171:Type Taxonomy for Author in fields 436
Figure 172: Structure of a CSP with four variables 455
Figure 173:Modeling of Distributed Cooperative Information Systems 479
Figure 174:Modeling_Unit structure 480
Figure 175:External and Temporal events 482
Figure 176:Event and state change 484
Figure 177:Meta proof techniques 491
Figure 178:Combining proof techniques 492
Figure 179:Modeling the world of Petri Net Formalisms 493
Figure 180:Broad overview of the system 508
Figure 181:Examples of unfilled frames 510
Figure 182:Example semantic net for concept classes 511
Figure 183:Parse tree for a typical query 513
Figure 184:Predicate-argument parse structure 516
Figure 185:Ambiguous nominals and extracted pairs 522
Figure 186:Overview of PMS 531
Figure 187:What is cfgpkg? 532
Figure 188:Timing diagrams for valid & invalid lock operations 538
Figure 189:Principles of merge and split 539

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Volume I

Distributed Intelligent Information Network

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1. OVERVIEW OF INFORMATION RETRIEVAL

1.1 Introduction.

Presently, in the times of high-speed communication and world-wide globalization of computer information, we realize the high demand for being able to inquire online information over a world-wide network in an intelligent style. Online information retrieval at any user site using an online display unit or a printer (or a teletype etc.) is a genuine requirement from information servers no matter from Internet, any other public network, private network or any commercial network.

Information retrieval (IR) is one automated online query answering system that accepts queries from valid users, searches its database, fetches detailed information and presents them on a screen as answers and may allow the user to navigate further using these information.

1.2 Educational information retrieval

Educational information is one representative application of such an online information access system. Online educational information is a wide-ranging need today whether at work-places, libraries, tutorial classes etc. Online educational information can be broadly categorized into two classes. 1) Bibliographic data, 2) Abstract & full-text of books, newspapers, reports, periodicals, news wires, dictionaries, encyclopedias, other monographs, course materials etc. The user may search bibliographic references on a particular topic, on a subject, on a few keywords, on a title, on an author's name etc.. But

also the user may directly go & search the abstract (or the full-text) of reference materials looking for a set of keywords or a concept. Nowadays searching a picture on the basis of a background, or a graph etc. are also being possible. Such an online information retrieval system may be used in libraries, educational institutions etc.. Further treatment of information retrieval will be in the light of text retrieval from references although there are other forms as question-answering etc..

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1.3 Features of expert searcher:

No matter if the information is bibliographic or full-text, they share some similarities in the technology for solution. As the volume of information available has grown, the users demand access to more information and then we want more specificity built into the searcher so that we are not overwhelmed by the explosion of unnecessary information. Some of the following behaviors are expected from a powerful text-searcher.

1) The system understands the concepts behind the user's inquiries.

2) The system is able to adjust the scope of the query (by using more specific or broader terms as necessary) depending upon the volume of the material retrieved.

3) The system tries to learn the subjective choices of the user & also follows the dynamics of the user's queries.

4) Search many repositories of information over a network parallel in an highly intelligent and coordinated way using a predefined protocol. Such coordinated searches will add precision to the retrieved results keeping network traffic under control and parallel searches will reduce total search time.

5) Filter out unnecessary information from the user that may overwhelm the user and present the filtered information in the ranked order of relevance.

These problems above involve sound reasoning and good knowledge representation involving application of methods of artificial intelligence and predictive statistical analysis. Enhanced retrieval capabilities are achieved through robust query processing, user modeling and inferencing methods. Many of these problems need to be solved heuristically or probabilistically because optimal solutions are not available.

As for application, such intelligent online IR system can easily be used in other various applications as medical diagnosis, online helpdesk, automated telephone answering, generalized image retrieval (multi-media) etc.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1.4 Overview of IR:

Information retrieval is a very broad area of research that have been continuing for at least last 40 years in various areas as information theory, syntactic analysis, semantic analysis, exact match of request, partial match of request, statistical analysis etc.. This complexity comes from the unstructured nature of the query as well as the text and also the semantics. Many a times a partial match of the request in the text may suffice. That is where deciding on the relevance of a document for a query becomes uncertain. Actually the references in the user's query should be used mostly by meaning, not by syntax, but detecting that is a very hard task. Also full semantic analysis of text in a complete structured style is a very hard subject and has not been achieved. This belongs to the area of modern linguistics and AI. At present various different models of analysis exists which will be talked about later.

The information repository to be searched is a very large database (at least a few Gigabytes). Alternately one may choose to create a document surrogate in the IR system consisting of the title, author, subject and abstract. But such surrogates can be viewed as documents and they form a very large database. Latency of such a large database access will be a major concern.

Today lots of commercial information retrieval services, information Database Gateway services, Internet search methodologies are existent. A networked IR system makes a better meaning for a library repository. A distributed networked IR system will generate many subqueries as a result of processing the user query. These subqueries are dispatched to the respective remote information servers. All the results are merged together and sorted by the order of their relevance (and filtered also). An IR system can be intelligent in applying decision rules, learn about the problem space, try to balance the load on the network or decide to route the query to an alternate node in case of failure of a network node.

1.5 Fundamental Issues:

The basic cornerstone of information retrieval research stands on the following three questions.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1) What is good document representation and how are these representations arranged? How can a concept representation be generated from a document?

2) How best can the user request be represented?

3) How can we compare these two representations to judge the likelihood that a document matches an information need?

1.6 Definitions

Query is the user's request which may be based on a language or may be structured using a graphical user interface. The criteria for the user's search may be varying in complexity from trivial as an exact match of word to word with wild characters to a complex criteria as keywords connected with many Boolean operators. The keywords in the query give the important concepts which may be words or phrases. Each keyword specification in the search criteria list of the query is a term. Some terms in the query may be based on a single keyword or syntactically longer than a keyword (offering a broader or more specific concept) formed using one or more relationships among a few keywords etc. (or it may be more complex than that e.g. about relationships among two documents and so on).

The keywords are normally extracted from a document and preserved in an index database which is searched for a query. The keywords are called the ***terms***. A ***conceptual model*** is the retrieval model that dictates the approach to handling the problem of IR providing the fundamental basis. ***Query operations*** are the operations performed on the user queries before a search begins. Various operations that are supported on the terms for query answering or at the time of index building are the ***term operations***. ***Document operations*** are the ones done when a new document is added to the database. ***Thesaurus*** is used to contain the other keywords connected to each keyword using the relationships as synonym, antonym, hypernym (super-ordinate), homonym, hyponym (sub-ordinate), meronym (part of another concept), holonym (superset of another concept) etc.. The most basic criteria for evaluating IR are ***Recall*** and ***Precision***. Recall is the ratio of no of relevant documents retrieved on a query to the total no of relevant documents existent in the database for the query. ***Precision*** is the ratio of the no of relevant

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

documents retrieved for a query to the total no of documents retrieved in the query. The overall characteristic function of an IR system may be formulated in terms of recall and precision ratio.

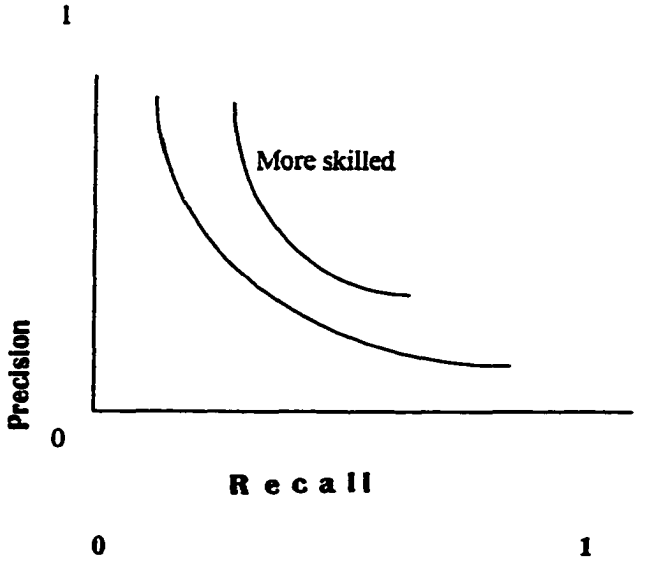


Figure 1:Variation of Recall ratio with precision of queries

These ratios are calculated over a long range of various queries. Fig 1 shows the typical variation of recall ratio against precision ratio.

Physical Aspect:

•**Data Structures:** Data structures which are commonly used for information storage are: flat file, inverted file, Signature file, Patricia trees, graphs, hashed files, semantic nets, conceptual graphs, case-frames, scripts etc..

•**Search Strategy Optimization:** In the physical level implementation of an optimized search strategy is related to the file organization, Thesaurus organization, determination of relevance-feedback. search algorithm etc..

•**Hardware:** The physical medium for information storage may be an optical disk for its high storage capacity. Large memory cache is helpful in searching the inverted file (keyword search). Parallel processor will improve the response time for searches for matching documents.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1.6.1 Conceptual Model:

The conceptual model forms the basis for any strategies of information retrieval. There are many different taxonomies of retrieval models but the most common one is used here that says there are two basic approaches to information retrieval. 1) Exact match of the user's specified criteria, 2) partial match of the user's specification (allowing some mismatches, some intervening words etc.). The exact match category follows the principle of looking for exact match of target text or target concept (including synonyms). The inexact match tolerates term-weights, broader match or narrower match of terms etc.. The strength of a match is determined by the semantic proximity of the retrieved concept for the query concept, percentage of syntactic matches etc..

One chosen taxonomy of the conceptual models for IR is given below in fig 2.

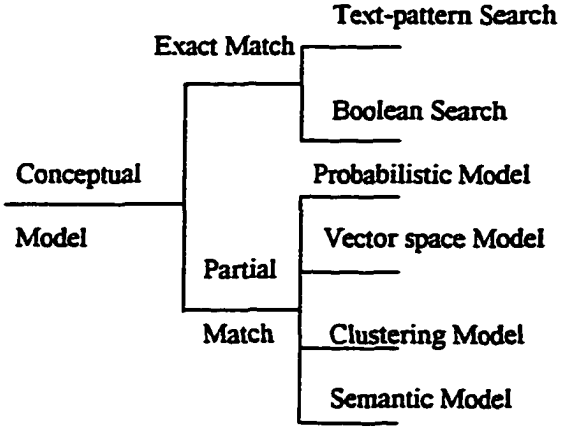


Figure 2:Taxonomy of conceptual model of IR

1.6.1.1 Text-pattern search model:

This exact match searches for one or more patterns directly in all the document texts instead of going through the Thesaurus or any semantic networks etc. Now a non-exact match can be implemented by allowing longer distances among the search terms on the text, or allowing partial matches. All the matching texts are displayed to the user in descending order of no of hits. Efficiency of such brute-force search methods depend mainly on text searching algorithms (e.g. using signature files or Patricia trees).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1.6.1.2 Boolean search model:

In this exact search model, the user's query is viewed as a Boolean expression of a few request concepts. The representation concepts of the documents that contain boolean term-weights are searched looking for the requested concepts and the Boolean expression of the result is evaluated. If the result is true, the document is chosen on the retrieval set else not. In this model all the term weights are boolean (1 or 0).This model can be extended to support some measure of relevance brought from the history and acquired through further interaction.

1.6.1.3 Vector Space Model:

In here, a document text is parsed & stored as a representation concept vector as $R=\{r_1,r_2,\ldots,r_k\}$ with r_i weight (a real-value) is assigned to term i. Similarly a user query is also a vector of terms. Many techniques have been used to automatically compute the weights in the documents, the most common being tf.idf weights which are based on the frequency of a term in a single document (tf) and inverse of its frequency in the entire collection (idf). Another choice for the term weight can be as $w_k = f_k . L_k$[30]. Here f_k is the frequency of the term in the given document and L_k is some type of term precision. Using the principle of minimizing the loss of information, retrieval of a document makes sense when the loss for rejecting a relevant document (P(rel).a2) (where a2 is the loss for rejecting a relevant document) is greater than the loss for retrieving a non-relevant document (1-P(rel)).a1 (where a1 is the loss for retrieving a non-relevant document). Using Bayes' theorem and the above principle the precision term can be expressed in terms of no of relevant documents with k terms assigned, total no of relevant documents, no of documents with k term assigned and total no of documents. Another way of measuring term weight is $w_i=f_i . DV_i$ where DV_i is the discrimination value. A user query is similarly analyzed into a request concept vector where each dimension specifies the strength of a single concept Similarities between the document representation vectors and request vector are measured using a similarity function. One typical similarity function is a dot product between two vectors. Finally, the representation vectors are ranked by the similarity values and a few top ones are retrieved.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1.6.1.4 Probabilistic Model:

This retrieval model takes a statistical analysis approach trying to maximize the probability of relevance between the query and the retrieved document. For this purpose, one tries to formulate some retrieval functions which can be used to rank the documents in the order of the probabilities. If F stands for the set of representations for information needs and D represents the set of document representation, then F X D represents the event space and the objective is to estimate the probability of relevance of a document-request pair i.e. estimating $P(R|d_i, f_j)$. One may assume some distribution of representation concepts in the documents and queries and then derive a ranking function to compute the probability . Probabilistic models support features of IR as term weighting, ranking and relevance feedback.

1.6.1.5 Clustering Model:

Cluster-based model is based upon the principle that similar documents will match similar queries. The documents are organized into groups called clusters based on the similarities of the documents. Each cluster is represented by a representative document called the centroid (say a concept vector). Now instead of searching each document individually, a large-grain level search is conducted using only the centroids. If a significant match is found, then a detailed match is pursued to generate the list of documents having relevance with the user request. Many algorithms and similarity functions exist to build clusters, cluster averages, matching user requests with cluster averages etc..

1.6.1.6 Semantic Model:

This represents documents by the semantics of the surrogates and the contents permitting a hierarchical structure. Through linguistic analysis of NLP, the concepts of interest in major subject areas are extracted from the full-text. The concepts are represented using some standard structures. The knowledge is established connecting objects of various classes formed using three principles of abstraction : 1) classification, 2) aggregation, 3) generalization. In Information retrieval the knowledge-base is traversed using some inference rules specifying how to traverse the knowledge-base in order to correlate query representation with document representation. Some models of the induction rules are: 1) activation method, 2) Bayesian inference method, 3) Dempster-Shafer belief theory etc..

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

An existing system normally is composed of a few of the above models at a time.

1.6.2 Data Structures & File Structures

To enhance the data access, keys and links to document data are stored using various data structures. In the simplest scenario, each document is stored as a text file and there are links in an index file pointing to the texts in the document files. Various text matching utilities (as grep, awk etc.) can be run to do the text matching on the index file. A few important data structures are described below.

1.6.2.1 Thesaurus:

This is a synonym dictionary which in all simplicity, specifies for each word-stem entry one or more synonym categories or concept classes; the ambiguous entries result in many-to-many relationships from word entries to concept classes. In further embellishments, it contains entries for other relationships (as meronym, holonym, hypernym, hyponym, antonym etc.) also. A hierarchical arrangement of terms or concepts, as in a library classification schedule, where for a given concept it is always possible to find a broader concept (more general) or narrower concept also may be included in the thesaurus.

A thesaurus should basically contain the technical words that determines the type of the document. Most of the times, the emphasis varies with the type of users. For example, *diodes* can come under

switching devices in engineering thesaurus, but will come under *computer components* in a computer thesaurus. In the initial level, the entries are arranged within concept classes. These concept classes may be some types of numbers (concept numbers). An example may be as follows:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

408	Dislocation	415	Cryogenic
	Junction		Cryotron
	Minority-Carrier		Persistent-Current
	P-N-P		Super-conduct
	N-P-N		
	Point-Contact	416	Relay
	Transistor		Switch
	Field Emission Transistor		
	Hole		

Figure 3:Simplistic Thesaurus with concept number

Although today lots of pre-built thesauruses exist for use, still some libraries may like to build its own automated thesaurus. For this it needs to scan the documents and extract important specific terms. For this very frequent words are skipped. Some very low-frequency words may have to be dropped to increase precision at the cost of recall. Once the word entries are chosen, the second problem is choosing the type of relationships permitted. Of course, some more general terms and some more specific terms should be allowed. But if very general synonyms (actually near-synonyms) are allowed for a term, then a query on that concept will result in a very high and overwhelming recall. If one goes down to very specific level

(say for example, *digital adaptive decision-feedback equalizer*) then there may not be a query on such a specific term and it may be useless from retrieval point of view. Also the recall may suffer since relevant matter is likely to be lost. The following principle of thesaurus construction may be used:

1. Very rare concepts should not be included in the thesaurus as they do not expect to produce many matches between the documents and search requests.

2. Very common high-frequency terms should be skipped from the thesaurus as they produce too many matches for effective retrieval.

3. Insignificant words should be studied carefully before they are added on the stop list because in some areas the insignificant word may become technically important.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4. Ambiguous terms may be included in the dictionary only for the senses where it is used in the document collections.

5. Each concept class should include only terms of equal frequency so that the matching characteristics are approximately the same for each term within a class.

1.6.2.2 Phrase Dictionary:

Such a dictionary contains all the most frequently used phrases that are found in common texts. An example is *"control systems"*. Such a dictionary makes the content analysis much more effective by offering index on an unambiguous phrase than on its individual ambiguous components.

1.6.2.3 Inverted File:

This file is some kind of an indexed file containing the word-stem terms of documents and pointers to the text. The structure of each entry is usually a keyword and a repeating group of document-id and byte offset. Keywords from the title can be left into another inverted file. Subject lists may be maintained in a separate inverted file. Various data structures may be used to implement inverted file as a sorted array (sorted by index), B^* tree [11], tries (recursive tree structures), hash structures etc. Patricia tree is a special application of trie.

1.6.2.4 Signature File:

Such a file contains profile signatures (bit-patterns) for the document. A signature is a condensed summary pattern (bit-pattern) for the content. A document may be represented by a signature or a series of signatures in various schemes. Sometimes the document is split into blocks (each of a fixed no of key words). Each word is hashed to generate a bit-pattern. All the signatures in a block are OR-ed together to generate a signature for the block. The document is stored as these signature blocks in the signature file. Searching is done by partial matching of the signature of the query with that of the document. On a partial match of signatures, detailed search can be continued on the selected documents.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1.6.2.5 Stop word list:

It is a list of words that are used extensively in texts, but they do not form keywords. Normally for word-stemming such words can be ignored. The examples are : a, an, the, but, although, prepositions, pronouns etc., nouns as machine, computer, equipment etc. Normally these words appear with very high frequency. All the stop words are normally maintained in a pre-built list.

1.6.2.6 Conceptual Graphs:

Such graphs are bipartite semantic graphs to represent knowledge of keywords and their relationships. When a complex concept of knowledge is built using a few interrelated keywords extracted from a text, they may be represented by such graphs. But to build such graphs, one needs the set of linguistic rules stated in some kind of a dictionary along with a dictionary pointing to such relationships. The advantage with such conceptual representation is that they can be more meaningfully handled (searched).

1.6.2.7 Hashed Functions:

Hashed functions and hashed files are used to represent keywords making it easy to search, useful in building the Thesaurus.

1.6.3 Query Operations:

The user's query can be entered on a graphical user's interface or using a query language. Also the answering can be through a long stepwise interactive session or it can be an isolated transaction. The most common operations on a new query are 1) query parsing, 2) building the keywords list in the query, the concept phrases in the query, 3) building a strategy for the search. Additional steps may be invoked in feedback systems as 1) each term is reweighted and additional terms are added internally to increase relevance[23].

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1.6.4 TERM OPERATIONS:

The term operations are the operations done on the keywords commonly at the search time or at the time of creating indexes on a new document. These operations may be classified as follows (see fig 3).

1) Stemming operation: This is to extract the morphological key from a keyword either in the query or in the text. Keywords are stored and compared in the word-stem form for extra retrieval effectiveness. In this way two keywords can match irrespective of their modifications thus rendering searches more effective. As example, if the query said *thorny*, it is more likely to find a match on *thorn*. There are a series of linguistic rules for stemming words. For example it may be the following:

   1) Check if the tail matches the standard list of suffixes as <noun>ies, <noun>s, <verb>ing, <verb>ed, verb>d etc.

   2) Check if the word matches with a final "*y*" changed to an "*i*" and a suffix added.

   3) Check if it matches a dictionary entry, with a final consonant doubled and a suffix added. The thesaurus may be maintained with the word-stems. Also all the standard suffixes may be maintained as a suffix list. A typical suffix list looks as in fig 4. It is visible that each possible suffix has a sequence no just like each entry into the thesaurus has a unique sequence no. Also along with sequence nos, there are syntactic codes which can be used later to recombine a word-stem with its suffix to build the acceptable words if needed for any reason say syntactic analysis. The syntactic codes in the suffix dictionary represent only partial homographs which must be combined with complementing codes attached to the word stems in order to determine which suffixes match with which stems. For example, a partial homograph such as 0t10 from the stem dictionary will combine with a partial homograph code from the suffix list, such as v00s0 to form a complete homograph. In this case the complete code is vt1s0, indicating a single object transitive verb in the third person singular. A typical suffix dictionary may contain about 200 entries in it.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Alphabetic suffix list		Syntactic suffix codes				
FICATION	058	058	NOUS			
FICATIONS	059	059	NOUP			
FIED	060	060	v00c0	p00 0	ADJ	
FIER	061	061	NOUS			
FIERS	062	062	NOUP			
FIES	063	063	v00s0			
FOLD	064	064	ADJ	NOVC		
FUL	065	065	ADJ	NOVC		
FULLY	066	066	av1			
FY	067	067	v00p0	i00 0		
FYING	068	068	r00 0	g00s0	novs	adj

Figure 4:Typical suffix table

2) Weighting operation: The indexing or query terms are assigned numerical values depending on

the statistical distribution of terms and distances among terms. These weights determine relevance on retrieval. There are various algorithms for determining weights.

3) Thesaurus operation: The Thesaurus is used to vary the scope of the query by broadening the scope of the query or narrowing it further down.

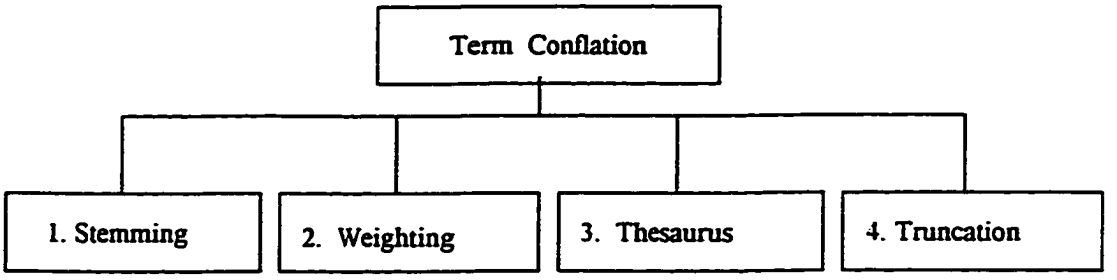


Figure 5:Various operations on terms

4) Truncation operation: This search-time operation interprets any wild character in the search term and matches the literal from the beginning or the end in the index.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1.6.5 Document Operations:

As a new document is added to the database, the indexes will have to be created so that the indexes point to the new document. In automated indexing scheme, this involves identifying major concepts and keywords in the abstract and the text, stemming on the identified keywords. This operation happens in three steps. 1) removal of very high frequency words and words on the stoplist (as they are found of no use). 2) suffix stripping. 3) detecting synonyms. In a probabilistic model the frequency of the important terms in the document and in the collection are also computed. These document operations can be manually performed also.

Document retrieval is another common document operation.

1.6.6 Relevance Feedback for term:

Each concept in the query is assigned a weight of relevance which determines the strength of the term in the entire query. This assignment may be internal (i.e. the first term gets highest value, second term next value etc.) or it may be prompted from the user. The knowledge of the prior queries of the user may feedback some factors in this relevance. That means the relevances are reweighted on basis of the knowledge of the previous queries to decide how strongly (or weakly) relevant the term is in executing the query. The new query is related to the entered query as $Q_1 = Q_0 + (1/n_1)\ \Sigma^{n1} i_{=1}\ R_i - (1/n2)\Sigma^{n2} i_{=1} S_i$ where $n_1 = n(R)$, $n_2 = n(S)$ $R = \{ R_1, R_2, R_3, \ldots, R_{n1} \}$, $S = \{S_1, S_2, S_3, \ldots, S_{n2}\}$. Now R_i is a concept vector of feedback (promotion) and S_i is a concept vector for exclusion (demotion) and the vectors are unit vectors.

So R is the promotion set and S is the demotion set.

1.6.7 Document Correlation

There are various mathematical functions to measure the similarities between two vectors (say a query concept vector and another document vector or two document concept vectors). Some major ones are Cosine correlation function which tries to compute the dot product between the unit vectors. This gives a cosine value normalized between 0 and 1. The formula is given by:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$\text{Cos}(q,d) = \frac{(\Sigma_{i=1}^{n} d_i q_i)}{(\Sigma_{i=1}^{n}(d_i^2).(\Sigma_{i=1}^{n}(q_i^2)))^{1/2}}$$

Here q and d are n-dimensional vectors for query q and document d. The overlap function is given by:

$$\text{Ovlap}(q,d) = (\Sigma_{i=1}^{n} \min(q_i,d_i)) / \min(\Sigma_{i=1}^{n} q_i, \Sigma_{i=1}^{n} di)$$

The Cosine function is more sensitive to the document length because of its denominator's dependence. The precision is maintained better in case of the Cosine function.

1.6.8 TERM-WEIGHTING:

There are terms found in the document . Given a term contained in the document, it is hard to say how far the document represents the term. Linguists take the approach of linguistic & semantic analysis. But enough sophistication in such analysis has not been achieved that can deduce a quantity of the representation. The popular methods are the statistical analysis. For example, there is a term k present in document i and w_{ik} is the weight. One possible formulae for $w_{ik} = f_{ik}/B_k$ where the numerator is the frequency of occurrence of term k in document i and the denominator is the total no of documents in the collection where the term k is found. This inverse document frequency postulates that a good term - one that should be assigned a high weight within a query or a document-displays a high occurrence frequency , while its document frequency is low. Another appropriate term-weighting function for term k is the term precision formula where $w_{ik} = f_{ik} . L_k$. Here the first factor is the term frequency and the second factor is the term precision representing the proportion of the relevant items in which the term occurs divided by the proportion of non-relevant items. In another formula the weight is given by term-discrimination value as $w_{ik}=f_{ik}. DV_k$ where the discrimination value (DV_k) is the difference between the two average space densities s_k' and s respectively, where s_k' is the space density factor for document pairs from which term k has been removed and s is the average space density for document pairs.

Using Utility measure theory[30] one can invent a utility function U(R,Q,T) that gives us utility of a system for retrieving r relevant documents and s non-relevant documents. In this function R is the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

total no of relevant documents in the collection, Q is the query and T is the retrieval threshold. The system retrieves all items whose similarity with the query is equal to or above the threshold T and the collection size is R+I where R is the relevant size and I is the non-relevant size. If the merit for retrieving each relevant item is v_1: s non-relevant documents are retieved at the penalty(cost) of c_1 (for each): c_2 is the penalty for missing to retrieve any relevant document (for each)and v_2 is the merit for not retrieving I-s non-relevant documents , then the total utility can be written as

$U(R,Q,T)=v_1 r - c_1 s - c_2(R-r) + v_2(I-s)$

If f(D,Q) is a random variable giving the cosine similarity function between any document D and the query Q, then

$$U(R,Q,T)= v_1 N P_r\{D \text{ relevant and } f(D,Q) >= T\} - c_1 N P_r\{D \text{ non-relevant and } f(D,Q) >= T\} - c_2 N P_r\{D \text{ relevant and } f(D,Q) < T\} + v_2 N P_r\{D \text{ non-relevant and } f(D,Q) < T\}$$

Where P_r denotes the probability of the event described in the braces. Now Pr {D relevant and f(D,Q) >=T} is given by P_r{D relevant}. Pr{f(D,Q) >=T|D relevant} and so are all the other probabilities above. Supposing the threshold variable T has a uniform distribution and using the fact that for a random variable Z if $P_r\{Z>= 0\}=1$, then $\int_0^\infty P_r\{Z>=z\}dz = E(Z)$ (the expected value of Z), integrating both the sides over dT, one obtains from the first part of the equation $\int_0^\infty U(R,Q,T)\, dT = v1\, NP_r\{D \text{ relevant}\}$.

$$\{\int_0^\infty P_r f(D,Q)|D \text{ rel}>=T\}\, dT - c_1 N P_r\{D \text{ non-relevant}\} \{\int_0^\infty P_r f(D,Q)|D \text{ non-rel}>=T\}\, dT$$

$$-c_2 N P_r\{D \text{ relevant}\} \{\int_0^\infty P_r f(D,Q)|D \text{ rel}<T\}\, dT + v_2 N P_r\{D \text{ non-rel}\} \{\int_0^\infty P_r f(D,Q)|D \text{ non-rel}<T\}\, dT$$

So LHS= $U(R,Q) = v_1 N P_r\{D \text{ rel}\}E[f(D,Q)|D \text{ rel}] - c_1 N P_r\{D \text{ non-rel}\} E[f(D,Q)|D \text{ non-rel}]$

$$- c_2 N P_r\{D \text{ rel}\}\{1 - E[f(D,Q)|D \text{ rel}] + v_2 N P_r\{D \text{ non-rel}\}\{1 - E[f(D,Q)|D \text{ non-rel}]$$

This gives $U(R,Q)= -c_2 NP_r\{D \text{ rel}\} + v_2 NP_r\{D \text{ non-rel}\} + (v_1 + c_2)N P_r\{D \text{ rel}\}E[f(D,Q)|D \text{ rel}] - (v_2 + c_1) N P_r\{D \text{ non-rel}\} E[f(D,Q)|D \text{ non-rel}]$.

Now setting $\alpha_r = P_r\{D \text{ rel}\}$, $\alpha_r = P_r\{D \text{ non-rel}\}$ and exchanging the cosine similarity formula for f(D,Q)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$f(D,Q) = \sum_{l=1}^{t} \frac{q_l}{(\sum_{j=1}^{t}(q_j)^2)^{1/2}} \cdot \frac{d_l}{(\sum_{j=1}^{t}(dj)^2)^{1/2}}$$

The new merit value is given by:

$$U(R,Q) = N \sum_{l=1}^{t} \frac{q_l}{(\sum_{j=1}^{t}(q_j)^2)^{1/2}} \cdot (\alpha_l - \beta_l) - c_2 N \alpha_r + v_2 N \alpha_r$$

where

$$\alpha_l = (v_1 + c_2)\, \alpha_r\, E\left[\frac{d}{(\sum_{j=1}^{t}(d_j)^2)^{1/2}} \,\middle|\, D_{rel} \right]$$

and

$$\beta_l = (v_2 + c_1)\, \alpha_r\, E\left[\frac{d}{(\sum_{j=1}^{t}(d_j)^2)^{1/2}} \,\middle|\, D_{non\text{-}rel} \right]$$

The equation for U(R,Q) is some type of a vector dot product ; so the optimum query, $Q_{opt} = (q_1, q_2, \ldots, q_t)$, will have term weights proportional to $(\alpha - \beta)\, K$ for some constant K. Hence the best weight w_k will be given as $w_l = (\alpha_l - \beta_l)\, K$ where K is some constant. The first factor is the difference between the expected values for term k in the relevant and the non-relevant documents, respectively. If one assumes that binary indexing is initially used for the documents and that the number of terms is approximately equal to L in each document, then one can write

$$E\left[\frac{d_l}{(\Sigma d_l^2)^{1/2}} \,\middle|\, D \text{ rel} \right] \cong \frac{1}{(L^2)^{1/2}} E[d_l \,|\, D \text{ rel}]$$

and

$$E\left[\frac{d_l}{(\Sigma d_l^2)^{1/2}} \,\middle|\, D \text{ non-rel} \right] \cong \frac{1}{(L^2)^{1/2}} E[d_l \,|\, D \text{ non-rel}]$$

Now substituting the expressions as follows:

$P_r\{D \text{ rel}\} = \alpha_R = R/N \qquad P_r\{D \text{ non-rel}\} = \alpha_{R'} = I/N$

$P_r\{d_l = 1 | D \text{ rel}\} = E[d_l | D \text{ rel}] = r_l/R$

$P_r\{d_l = 1 | D \text{ non-rel}\} = E[d_l | D \text{ non-rel}] = s_l/I$

The optimum term weight of expression then finally becomes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$w_l = (v_1 + c_2) \cdot \frac{r_l}{N(L)^{1/2}} - (v_2 + c_1) \frac{f_l - r_l}{N(L)^{1/2}}$$

where $f_l = r_l + s_l$ represents the document frequency of term l- that is the no of documents to which term l is assigned. After dropping the subscript l for r_l and f_l the expression may be written as

$$w_l = \frac{1}{N \cdot \sqrt{L}} [(v_1 + c_2 + v_2 + c_1)r - (v_2 + c_1)f]$$

The two characteristics stated below can be easily verified about f and r.

1) r increases monotonically with increasing f; that is, the more documents contain a given term, the more

relevant documents will contain the term.

2) As f increases, the rate of increase of r decreases; that is, when f is small, the proportion of relevant documents containing a given term is relatively larger than when f is large.

Hence this suggests a functional relationship between f and r as $r = a + b \log_2 f$ where a and b are constants. The equation above for w_l represents am increasing function of f for small f upto the maximum point $f=[(v_1+c_2+v_2+c_1)/(v_2+c_1)](\log_2 e)b$. After that, as f increases further, w_l is a decreasing function of f.

1.6.9 Scope Adjustment:

Adjustment of the scope of the query may be essential when the retrieval set is unusually small or too large. When the size of the retrieval set crosses certain preset limits, such a step becomes necessary. If the set is too large then the high weighted keywords may need to be more specific. The Thesaurus is looked up for finding narrower (or hyponyms) words. Specially the ones that are narrower than a few specified keywords will be more useful. Another approach is to readjust the weights after adding more keywords internally. Relevance feedback will be required for this. To adjust the scope, one or two further trials may be essential.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1.6.10 Semantic Modeling:

Apart from the four previous models of information representation and retrieval, there is a new emerging model which bases its analysis upon the semantics content. In this model , the concepts are extracted from the documents with the semantics intact and stored in some knowledge representation form as semantic network. So the keywords are searched for and then the total clause or phrase is passed through the semantics analyzer. The semantics-analyzer analyzes all the phrases and looks for some common patterns of linguistic structures based upon the keywords. Then applying some preset linguistic templates as *subject-verb-object* or *subject-predicate* etc. it tries to extract the significant concepts and stores that as a semantic data-structure. Semantic networks or conceptual graphs[10] are some typical examples of knowledge-representation. A similar operation (concept extraction) is carried out during query-inputting.

In the world of pure semantic model, the semantic structures of clause-concepts can be further bound together on the level of a sentential semantics. The sentential semantics can be sequenced together on the level of a paragraph semantics[26]. These representations can be contained in another structure called chapter semantics and those in turn can be bound in the level of topic semantics. Subject semantics can be derived analyzing the topic semantics. Such a pure semantic analysis is a far-fetched concept today due to its complexity in memory and run-time. But a phrase-level or clause-level semantic structure is possible to be analyzed. The advantage of such representation is better recall and improved precision. The better recall comes from the texts that are semantically equivalent, but literally different. Such texts are also retrieved due to semantics analysis. The better precision comes because the presence of the concept is looked for, not just the keywords. So the text will tend to suit the query better. During information retrieval the knowledge-base is traversed using some *inference rules* (specifying how to traverse) in order to correlate query representation with document representation. Some models of the induction rules are 1) *activation method*, 2) *Bayesian inference method*, 3) *Dempster-Shafer belief theory* etc..

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Natural Language Processing (NLP) based semantic model is a popular trend today. This model extracts and represents the semantics. Such models are commonly used in very specific domain of problems.

1.6.11 Linguistic Model:

Although linguistics is a broad field of language including sound mechanics and phonetics, but still syntax and semantics of natural languages are the prime concerns of the linguists. As document indexing and query answering is involved with language processing, hence the experience of linguistic research helps tremendously in information retrieval. For parsing a sentence in natural language, one may use a grammar of any two kinds. 1) phrase-structure grammar and 2) transformational grammar. The phrase-structure grammar gives the breakdown of a sentence in its phrases and various predicates. For example,

S → NP + VP NP → Art + Noun VP → Verb + NP Art → The|a|an Noun → Dog|Cat

Verb → chased Now a phrase-structure grammar is not enough to interpret the meaning of a sentence because the same parse-tree can lead to many interpretations. A transformational grammar places some semantics within the parse-tree. Those semantic nodes are so-called preterminals and form part of the base-tree. For a sentence as '*The man offered some flowers to the children*' a parse-tree with such preterminals, will appear as follows:

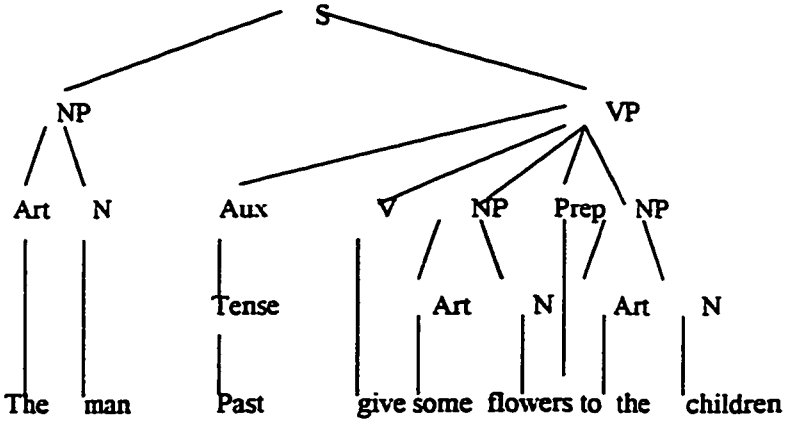


Figure 6:deep structure on phrase-structure grammar

The preterminals read as: **Art N Tense V Art N Prep Art N** and they add some kind of a deep

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

structure over the phrase-structure grammar. This deep structures are related to the surface structures by the rules of the transformational grammar. Each of these rules takes a complete tree as the input and delivers a complete tree as output. Some transformational grammars specify roles and thus add semantics in terms of objects as actors, instruments, operatives, modifier, location, time, description etc. Approaches to derive the deep structure are many as 1)approaches using rule form, 2) approaches using deep-surface distinction, approaches starting from surface structure without going through the intermediary structures. Such an in-depth syntax (partly syntactic) analysis is useful in identifying structural relations in the actual index descriptions and also to identify term relationships in a query concept. If indexes are stored with their deep structures, then searching semantically equivalent concepts becomes easy[32]. Thus linguistic analysis finds wide application in indexing and query retrieval where concepts are formed of many associated terms.

1.6.12 Object-Oriented Model:

One common way of arranging the concepts in the index is by identifying the keyword objects, and classifying them in a hierarchy of specificity. In this way, the index will look as a forest of hierarchic class trees. The keywords will represent some objects (mostly nouns) and adjectives, adverbs will be qualifying them. Thus if there is a key phrase as *"American Civil War"*, the class tree can look as

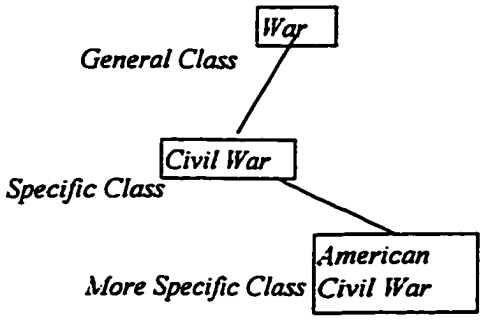


Figure 7:Object classification

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Such classification of keywords makes the search fast, because in the beginning, one needs to search the roots only. The knowledge is organized using the three principles of abstraction classification: 1) classifying, 2) aggregation (relationship among classes), 3) generalization (forming class hierarchy).

1.6.13 Syntagmatic model:

A new model for document analysis that has emerged as a result of a merger between linguistic syntax-analysis and paradigmatic analysis is syntagmatic model[27][28]. Here the knowledge of the concepts forms the index and they are stored as frames. A frame represents more a scenario than an individual object. This means a concept is shown more as interaction of a few objects. The individuals involved in this interaction are contained as various slots in the frame. These slots can contain an object, reference to another object or reference to another frame. The different roles are marked by the different slots. This modeling is better where queries refer to an action or a scenario rather than an object in reality. These frames can be arranged in a hierarchy to show when a concept is a subset of another broader concept. All types of relations (*is-a-kind-of, part-of, antonym, correlated-with, more-specifics-are* etc.) are still possible between two frames. A frame is identified chiefly by the activity (action involved), some follow-up adjectives and then a linguistic template (which broadly identifies the important role-players). For retrieval , queries are also converted into action-frames and then searched through the index frames. Such frame-based information retrieval improves recall as well as precision. It gives better precision, because it captures various detailed-aspects of the activity and compares all of them. It gives better recall because it can adjust the scopes by moving through the hierarchy. A taxonomy of today's research in IR is provided in fig 8 below.

All the models above, the semantic model, o-o model, syntagmatic model are derived from the field of AI. The *Bayesian inference method* uses a belief theory in attempt to maximize the relevance of the retrieved document set to the query. This inference model builds two networks 1) object network which is made of all the documents and the concept nodes and 2) query network which is made of all the query representations and user's information need. Existence of a node p means propositional variable p

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

may be true. An arc pointing from node p to node q means there is a possibility that q may be true given p is true. The estimated belief is given by P(q|p) means the strength of q being true if p is true. If many arcs are pointing to the same q from different parents, then q carries many beliefs as P(q|p1), P(q|p2), P(q|p3) ... etc.. Finally the whole composite inference network is built and those documents are chosen which have high belief to the user information need.

The Dempster-Shafer theory of evidence, also can be used as an alternative method for evaluating these kinds of probabilistic inference networks. Here rather than computing the belief associated with a query given a set of evidence, this method computes the probability that the evidence will allow us to prove the query[25].

Also other frameworks as modal logic, predictive recursive retrieval runs, knowledge-based search tactics etc. have been used for document retrieval. Some learning models have also been tried in order to come up with the best estimation function for P(q|p) including use of neural networks, genetic algorithms etc.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

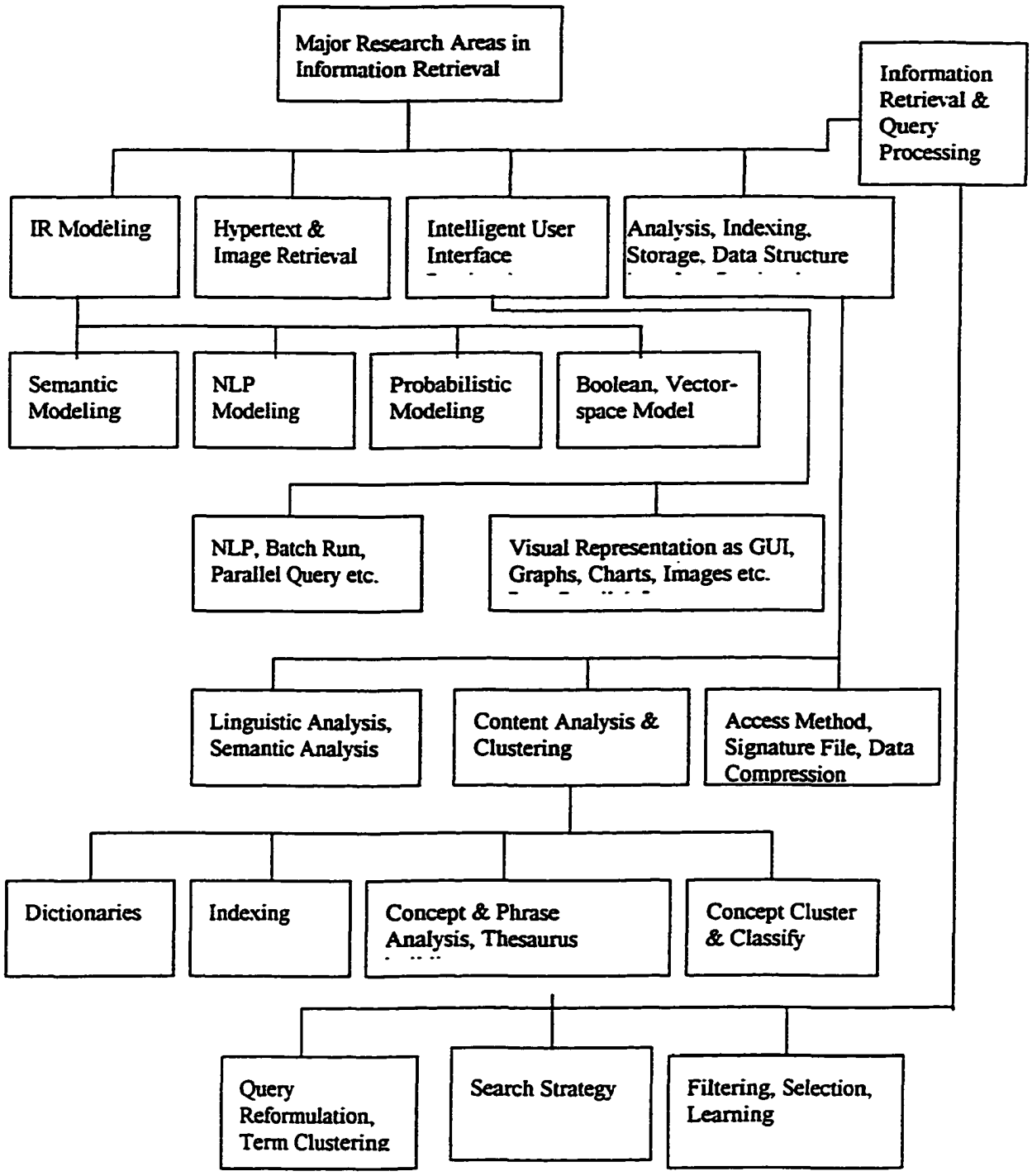


Figure 8:Taxonomy of today's research areas in IR

Content Analysis, Indexing, Data Structures, Storage is an important field of research today specially when automated classification and indexing becomes a requirement. Content analysis can be linguistic or

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

probabilistic or on a vector model. Initially terms are extracted from a text, conflated and then multi-word phrases or keyword concepts are coined and fed into index. If the system is using any knowledge-base then

additional relationships are also established. In case of a semantic model, the semantics-content is extracted, classified and stored as indexes. Also various data structures, access methods, data compression algorithms are concerns as a part of the internal schema.

Information Retrieval and Query processing is a very wide field of active research today. Query reformulation involves expanding a user query internally using relevance feedback, adjusting the weights on the terms in the queries, conflating the terms, parsing query and determining a strategy. During this process, the query terms may be clustered on a basis of similarity (and the centroid concept is retained), User profile may be used to provide a relevance feedback. Search strategies, use of learning models (including neural net, genetic algorithm etc.) , ranking methodology are also concerns of this area.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2. AUTOMATED INFORMATION INDEXING AND RETRIEVAL

2.1 Introduction:

As one can easily see that the importance of the initial information analysis and classification is paramount considering all later search and retrieval operations depend on it. The topic of information analysis is embedded within the generation and use of dictionaries containing the technical terms, synonyms

and cross-references in topic hierarchies. The important things to examine are the various principles of dictionary construction, automatic classification procedures. But first of all the principles of language analysis should be reviewed.

2.2 Language Analysis:

In today's information retrieval systems, automated indexing forms the most important part and very simplistically speaking the task of automated indexing means accepting a document with natural language text and invoking some automatic procedure to generate content identifications and content representations. The complexity of the language and the various syntactic and semantic structure causes the immediate problems. In order to begin effective language analysis the following guidelines may be remembered:

1. Words that fill out syntactically but do not increase the information content of the sentence must be ignored (e.g. *may*, *am* etc.).

2. Synonymous or related words or expressions must be recognized for an accurate content analysis.

3. The same word may be used to mean different sense in different contexts. It is important to identify such homographs and recognize the proper meanings.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4. Many syntactic equivalent constructions are possible to imply the same idea which provides an extension of the synonym concept (like a paraphrase). A few major types should be known.

5. The use of indirect references is common in natural language, where pronouns, collective names and other predicates are used to refer to entities known from previous contexts. The proper identification of the antecedents is very hard specially when there exists ambiguity which needs to be resolved with world-knowledge or knowledge of the domain.

6. There may be associations between two words which are not explicitly specified, but can be deduced from the context or from other texts analyzed in past. The identification of such relations requires deductive reasoning.

7. The meaning of many words may change with time and so it is advisable to update the dictionary in time.

As text is treated for content analysis, the analysis system should provide consistent language normalization which is achieved using a consistent dictionary. The following types of dictionaries are quite familiar in this context:

1. A negative dictionary contains the stop-words.

2. A thesaurus contains for each word entry the corresponding synonym category or the concept class.

3. A phrase dictionary contains very frequently used phrases (concept combinations). Phrases treated as a whole gives better content identification than the sum of the often-ambiguous components and hence the phrase dictionary is used to contain the useful phrases. One example is *data structures* or *programming standards.*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Individually analyzing, the component words *data* and *structures* do not provide very specific information. But the phrase implies representation and format of the data.

4. A hierarchical classification of the concepts that makes the scope adjustment possible.

2.3 Dictionary Construction

Thesaurus: Thesaurus is a series of grouping of words (or word stems) into subject categories called concept classes. Each concept class is uniquely identified by a 3-digit concept sequence number. A common thesaurus may look as follows:

```
408 Switching-device
      Junction
      Minority-carrier
      Majority-carrier
      Diode

415 Cryogenic
      Cryotron
      Super-conduct
```

Figure 9:Thesaurus excerpt in concept no order

Word-stem	Concept-no	Syntax code
Borrow	28	043
Brain	404 0235	070
Brief	32232	001043071

Figure 10:Thesaurus excerpt in alphabetical order

When constructing a thesaurus, there three questions that need to be resolved. 1) which words need to be included, 2) what concept classes should be used, 3) given a word, where should the category be placed in the dictionary?

Once a document is scanned, the individual words are identified along with their frequencies. Stop-words, non-informative words are dropped. Now the keywords can be extracted from the title, or title and section headings, or the abstract or the entire text. Regarding the very frequent words, some are non-technical and so can be dropped right away, but some may be technical with respect to the subject. In case they are meaningful, still most documents will carry it and so these keywords hurt discriminating power. Thus these words may increase recall but reduce precision. A compromise must be met between high recall and

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

lost precision. There is a similar problem with low frequency words. For example, if *Morse code* (a low frequency word) is included in a category, then other coding systems also need to be included. Then a search on *Morse code* may also retrieve documents dealing with coding systems (a more general term) but not with this specific system. Once again a compromise must be met and some words must be dropped.

Once the words are chosen, the next problem is choosing the categories. This is a fuzzy question, because if very broad categories are included, then a query on such term will overwhelm the user with little precision. On the other hand, if the concept classes are very specific, then it may be hard to retrieve partly related texts and recall may suffer. Specifically high-frequency terms can not be mixed with low-frequency terms because then specific terms will lose its effectiveness when they are considered equivalent to broad terms.

Word Stem Thesaurus: In earlier times, the searchers were interested in word match rather than concept match. Searchers were interested in the informative technical part of the word rather than along with the exact temporal tense as well as the plural or singular representation as in the text. Hence the words needs to be normalized (called stemming) ridding of affixes (suffixes and prefixes). In the word-stem thesaurus, the stem, suffix, sequence number and the frequency are stored. The advantage is measuring word-similarities becomes easier. For two words to be similar one has to have the other stem and the bigger one must be a compound word. Using various stemming algorithms, word-stems are extracted and stored in the thesaurus.

Phrase dictionary: This dictionary contains standard high-frequency phrases and assigns them a concept class. An example will be *Phantom of Opera.* A phrase dictionary may have word-stems or it may be built upon concept numbers. Two principal methods are used for phrase detection. 1) The *statistical phrase* dictionary is used to confirm the statistical co-occurrence of all the component terms. 2) The *syntactic phrase* dictionary includes not only the components but also the syntactic dependency. Thus if the dictionary contains a phrase as *rewriting rules,* then probably the dictionary will contain it with the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

word-stems in the form *rewrite rule* where the first concept depends on the second. Now this document will satisfy a query with terms as *rewritable rules* but not as *rule-based rewritable systems*. That is because the syntactic dependency is not satisfied. Excerpt of a statistical phrase dictionary may look as follows in fig 11. Phrase concept=543 component concept1=544 component concept2=608 component concept3=0

component concept4=0 component concept5=0 component concept6=0

The syntactic phrase dictionary has a complex structure made of component concepts, the syntactic indicators and syntactic relations that may hold between the concepts. Fig 11 shows an excerpt of a criterion tree (syntactic tree).

Name of Tree	Output Concept	First node Concept	Second node Concept	Type 7 Serial	Type 15 Serial	Type 16 Serial 399
MAGSWI=422		(185,624) /	(225)			$7/143,15/398,16+
MCHBKD=69	(689)/(600)$1/148					
MTHMOD=722	(1273)/(116)$7/172					
NOGSIM=679	(353)/(604)$7/180					

Figure 11:Excerpt of a syntactic tree

The additional spaces have been added for clarity of view only. In the example, the first phrase carries concept no 422 and it has concept indicator of MAGSWI (to indicate that it has something to do with the magnetic switches). The first component concept can be either 185 or 624 and the second component is concept 225. The indicators after the $ symbol indicate syntactic information. This means the phrase must be of syntactic type 7 or 15 or 16. There are four main classes of syntactic specifications, among noun phrases, subject-verb relations, verb-object relations and subject-object relations. The syntactic classes are again subdivided into approximately 20 syntactic types each of which specifies a particular syntactic relation between the components. The syntactic relations that shows for SYNTAX is shown in fig 12 that describes the allowed syntactic tree. In this new example, the first concept is 11 or 158

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

whereas the second concept is 102 or 188 or 170. Also the allowed format types are 1,3,4 and 13. These formats are drawn as trees in the format column (syntactic dependency tree).

Phrase specification:

SYNTAX (11,158)/(102,188,170)$1,3,4,13

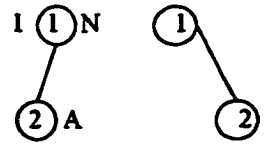
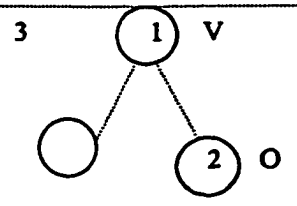
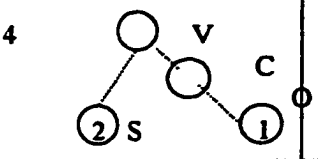
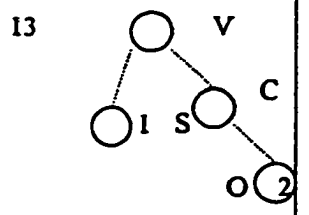
Concept Node 1	Concept Node 2	Formats	Sample Phrases
11 ANAL SYNTHESIS SYNTHES SYNTHET	102 INTERLINGU LANGUAGE		1 SYNTACTIC ANALYSIS PHRASE RELATIONS ANALYSIS OF SENTENCE
	170 PHRASE SENTENCE		3 WE CAN ANALYZE THE LANGUAGE ..SYNTHESIZE A SYNTAX
158 CLASS CORRESPOND GROUP INDEPEND RELATE	SUBJECT WORD 188 GRAMMAR SYNTAX SYNTACTIC		4 THE GRAMMAR IS NOW AVAILABLE FOR ANALYSIS
			13 THIS ANALYSIS IS APPLICABLE TO RUSSIAN GRAMMAR

Figure 12:Criterion phrase description

Dependency trees are characterized by the fact that vertical displacement along a given path of the tree denotes syntactic dependence. The dependent structure comes under the governing syntax node. For example, format type 1 specifies that the second component, corresponding to concept number 102, 88 or 170, be syntactically dependent on the first component corresponding to concept 11 or 158. In addition the second component is an adjective where the first component is a noun. Example phrases corresponding to the syntactic structures are given on the right side. Due to multiple assignment of concepts to phrase

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

components, multiplicity of syntactic format types for each phrase, a given syntactic phrase can match hundreds of English phrases or sentences. This feature matches different phrases that are semantically similar but syntactically different. Since the syntactic dependency is directed from a dependent component to a governing component, the grammatical structure of a syntactic phrase is determined. Thus using format type 1 the phrase *"phrase relations"* is acceptable, but *"relational phrase"* is not acceptable.

2.4 Concept Hierarchy Dictionary

Hierarchical arrangement of subject headings that have been long used in library science makes it easy to classify more specific topics under more general ones. In this way, one can start from the topic of *Computer Science*, then *Algorithms*, then *Graph theory*, then *Trees*, and finally *Transformational trees*. Or one can start from *Transformational trees* and then go all the way up to *Computer Science*. A similar concept hierarchy dictionary is a common idea in indexing. This hierarchy tree arranges the concept numbers (as gotten from the thesaurus) in this hierarchy using the inclusion relationship. Excerpt of such a concept hierarchy is shown in fig 13.

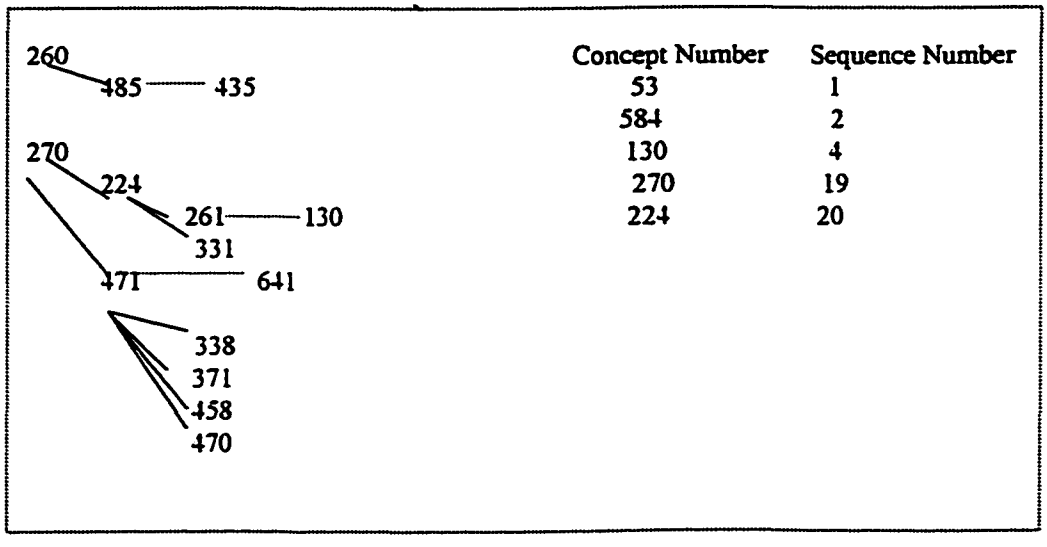


Figure 13:Concept hierarchy

Here The solid lines refer to the hierarchy relationship and the dashed lines indicate cross references.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

For example, concept 485 is specific within a broader concept 260. The cross-references indicate *see also* types of entries in Roget's thesaurus. Specifically speaking, the highly frequent concepts (with frequency of high-occurrence) should be placed higher in the list if sequential search procedure is used.

2.5 Dictionary Performance

The relative effectiveness of the dictionaries become evident studying some experimental results based on the averages of 17 search queries. Each data is measured as the average results of the search queries run over a collection of 500 documents. The recall-precision graphs reflect the performance comparison for the various dictionaries. The graphs are drawn by measuring recall and precision at different recall points. For example, if documents are retrieved in descending order of correlation with query, recall may be computed after retrieving five documents, again after ten documents and so on. Down each case, the recall increases (from 5 to 10 etc.) as more relevant documents are retrieved and possibly at the same time precision falls.

2.5.1 The Stem Thesaurus

This thesaurus is used as part of a word matching or word-stem matching procedure. It is possible to use it for all the word-stems in the document or only the word-stems from titles or from titles and section headings. Also a given sequence number from the stem thesaurus can be assigned to a document specification with a uniform weight if, the word-stem is found in the given document. Alternatively, the sequence numbers can be weighted to reflect its frequency of occurrence in the document. Fig 14 illustrates the recall-precision performance of unweighted indexing and fig 15 illustrates the performance for weighted indexing.. In fig 14, one curve is for word-stems in title, and the other one is where word-stems are extracted from the document text only.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

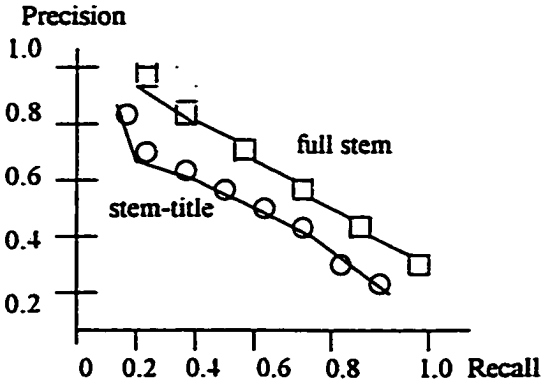


Figure 14:Performance for unweighted indexing

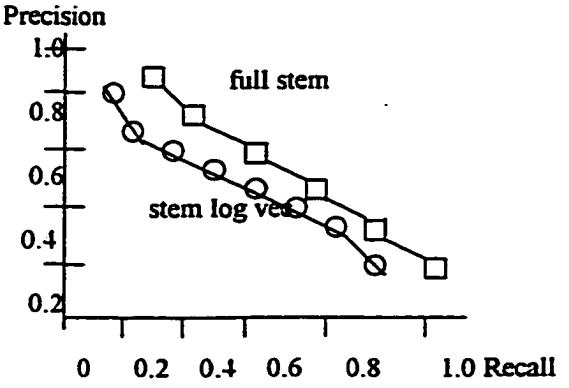


Figure 15:Performance for weighted indexing

As can be seen, retrieval using word-stem thesaurus for the document titles only performs inferior than when applied to full document-text. Fig 15 shows the comparison for unweighted indexing (stem logvec) against weighted indexing on a full stem document. It is clear that weighted indexing over full-document word-stems is most useful.

2.5.2 The regular thesaurus

The regular thesaurus offers synonym recognition and thus can offer better recall by including synonyms or other related words. Comparing the performance of a synonym dictionary with a word-stem dictionary one can say, at the low recall level the word-stem thesaurus gives better precision than a synonym thesaurus due to false retrievals (with synonyms). But at higher recall, the synonym dictionary gives better precision.

2.5.3 The phrase dictionary

The statistical phrase dictionaries were more popular because of the ease to build them. Comparing the performance, a statistical phrase-dictionary gives a boost to the precision of the thesaurus. When the concepts of the phrase dictionary is given higher weightage (say 1.5 times) than the individual concepts in the regular thesaurus, the precision improves further.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

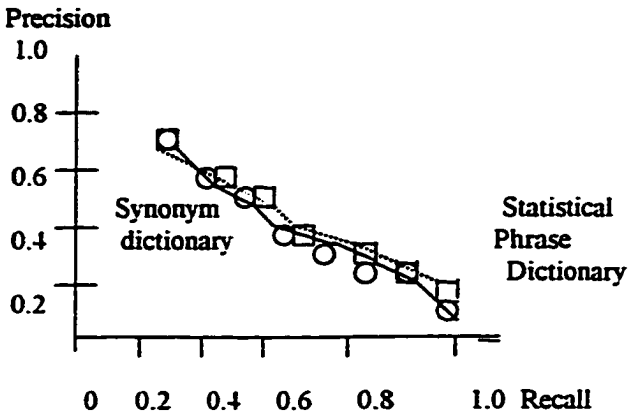


Figure 16:Comparison based on phrase dictionary

Fig 16 shows a comparison of the performance of synonym dictionary and a combination of synonym and phrase dictionary. At low recall the precision goes low due to misfiring (more so because it is statistical) of the phrase dictionary, but at higher recall the precision improves for the combination. Even higher weights improves the performance further.

2.6 Automatic Thesaurus Construction

Construction of a manual thesaurus is always a slow process due to many people that are involved, also it requires much training and finally it is partly subjective and hence inconsistent. In this perspective, an automatic thesaurus will be better suited, because the thesaurus can be easily extended to cover new subjects and it is possible to investigate better algorithms for building it. There are some fully automatic methods, some semiautomatic methods.

2.6.1 Fully automatic method

Most automatic methods for dictionary construction are based on the vocabulary contained in a sample document that is typical for a given subject. Given a source document, a frequency count is made of the terms (or word-stems) as chosen from the source document using the rules of frequency characteristics. The frequency of the terms assigned to the documents are arranged in a term-document matrix or a term-document graph. A typical term-document matrix is shown in fig 17 that gives the frequency of terms assigned to documents where the columns represent the standard terms and the rows represent the documents.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

	T1	T2	T3	T4	T5	T6	T7.....
D1	3	0	0	2	0	6	1
D2	0	0	1	3	2	0	2
D3	0	2	3	0	4	0	0
D4	1	2	1	0	3	1	0

Figure 17:Term-Document Matrix

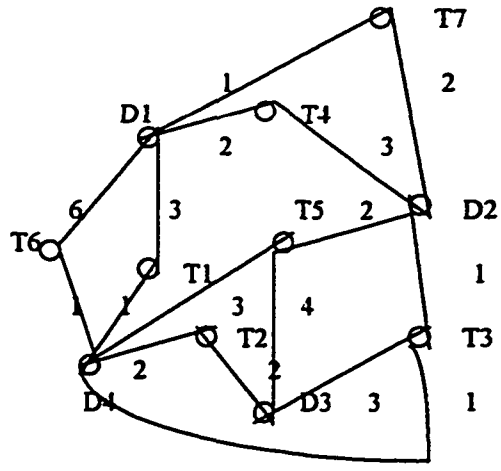


Figure 18:term-Document Graph

Above fig 18 shows the term-document graph for the same term-document matrix. Element (i, j) in the matrix represents the weight for term Tj assigned to document Di. Given such a matrix or graph, using the co-occurrence characteristics of the terms one can calculate the similarity coefficient between two given terms. The similarity coefficient between a pair of terms depends on the frequency with which the terms are jointly assigned to the documents in collection. For example, terms T1 and T6 are both assigned to documents D1 and D4 and void from documents D2 and D3. Hence, the term association process can group these two terms together in one concept category in the thesaurus. Proceeding this way, the association process may come up with three association groups, one made of terms T1 and T6 (because of joint assignment to documents D1 and D4), terms T7 and T4 (because of joint assignment to documents D1 and D2), and finally, terms T2, T3 and T5 (due to joint assignment to D3 and D4).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

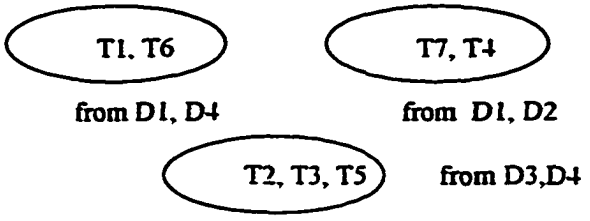


Figure 19: Term groupings for Dictionary

2.6.2 Semiautomatic methods

It may be argued that the above principle of grouping on the basis of similar patterns of assignments do not hold true always. For example, a word may be ambiguous and so a gross frequency will not reflect all the senses. Thus human intervention may be necessary to apply the association aid and decide to group terms together. The user may be given a concordance which is a list of all the occurrences of the terms along with some short contextual text. When the list of the terms are available to the user, they may be grouped in various ways:

1. The user may informally verify each pair and decide whether to group them or not.

2. A set of syntactic frames are given by the user specifying a term P can be replaced by a term Q in a context X. The grouping can become mechanized after this point.

3. A set questions may be asked to the user about a term's properties (e.g. is this a material body or a concept or pure abstraction? Is this about a class of items or a single item? If it is a material object then is it a live object or dead ? etc.). At the end of these questions, the term should have moved down the decision tree somewhere and that is its group. At the end a term-property matrix may be built. Once such a term-property matrix is built, it may need to be modified further. Of course, the terms that have common assignments will easily group together. But if groups become too large, a new property may have to be added or if groups are too small, then some properties will have to be dropped so that some terms can be coalesced together.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.7 Automatic Hierarchy Arrangement

Even after the groups of terms are formed for the Thesaurus, the hierarchical arrangement of the concept classes have to be built. Even within the groups of association among terms, unequal frequency of words need to be arranged in such a way that they may be broken hierarchically so that more frequent words form categories and less frequent words form sub-categories. If the broad concepts can be placed higher in the tree and specific concepts can be placed lower on the tree, then it should follow the frequency characteristics naturally. There are a few semiautomatic methods for this classification.

1. In this method, the questions used for thesaurus construction (as mentioned before) are used and terms are classified depending on the answers. Fig 20 shows such a hierarchy arrangement tree.

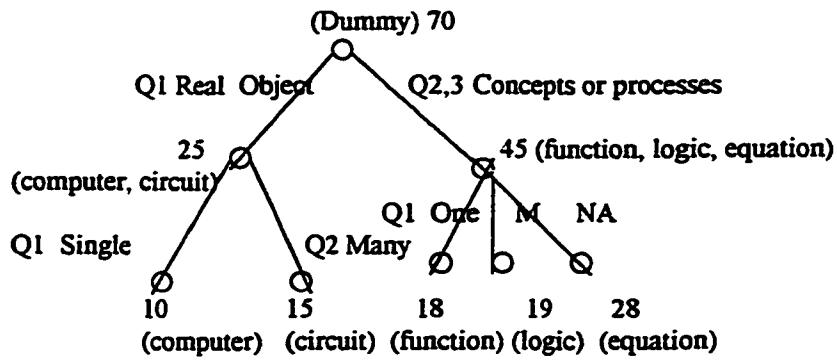


Figure 20:Hierarchy arrangement

The vocabulary set applied in this simple example is the following:

computer: frequency=10, circuit:15, function:18, logic:19, equation:28

The first question applied is whether it is a physical object or concepts or processes? The result leads to forming two groups called "physical objects" and " concepts and processes" with frequency of 25 and 45. The 3 questions go on this way leading to the five concept classes and other broad concepts.

2. This method works differently by starting on the word frequency list but not on the thesaurus groupings. The hierarchy is constructed first and the thesaurus is later built. The two way path from the dummy root is followed by letting the highest frequency term T_i and the other words like it form a node

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and the rest terms form the other node. Again the node with T_i term is chosen and broken into two nodes , one with T_i and very similar words and the other node with the rest. If the number of words in a node is reasonably small, progress stops there and other intermediate nodes are split. At the end, the leaf nodes provide the groups and intermediate nodes give the broader classes and the tree becomes the hierarchy. While partitioning always the word with highest frequency and some like it go on the left branch and the rest go on the right branch.

3. This method uses the term-property matrix which is built manually. The rest part of the hierarchy construction is automatic[33]. Suppose, there are two terms X and Y in the document for which the property vectors are being compared. The following situations may arise:

a) Terms X and Y carry different properties and hence are not related.

b) Terms X and Y carry the same properties and the property weights are very similar so that no one dominates the other.

c) Terms X and Y carry the same properties, but the weights of X are mostly stronger than Y, then X dominates Y or X should go higher above Y in the hierarchy.

d) Terms X and Y are related i.e. Y dominates X.

It is easier to compare the vectors by using a similarity coefficient function and possibly an asymmetric

function may be more preferable. Given two k-dimensional property vectors x^i and x^j for two terms T_i and T_j a possible similarity function can be

$$c_{ij} = \frac{\Sigma_k \min (x_k^i, x_k^j)}{\Sigma_k x_k^i}$$

Now a term-term relationship matrix can be constructed where each element will be c_{ij}. One may note that this coefficient varies between 0 and 1 where 1 means the property vectors are identical. A cutoff value K may be already agreed upon and a hierarchy can be built using the following algorithm:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

If c_{ij} and c_{ji} are both below the cutoff value K, then terms i and j are unrelated.

If c_{ij} and c_{ji} are both above the cutoff, then term i and j are in the same concept class.

If c_{ij} is higher than c_{ji}, then term j is broader than term i.

If c_{ij} is less than c_{ji}, then term i is broader than term j.

The choice of K makes a difference in generating an effective hierarchy. If K is very high (close to 1), most of the terms may become unrelated. If K is too small (close to 0) then most of the terms will come in one group. Now the coefficients c_{ij} can be arranged in a matrix showing all the term-term similarities. Then a reduced similarity matrix can be built where entries are 0 or 1 (depending on if c_{ij} is less than K or not).

Fig 21 shows those two typical matrices. From the reduced matrix and the above rules, the resulting hierarchy is built. The result is actually a mesh rather than a tree.

	T1	T2	T3	T4	T5
T1	1	1/9	8/9	5/9	5/9
T2	1/5	1	2/5	2/5	4/5
T3	8/10	2/10	1	7/10	7/10
T4	5/7	2/7	1	1	6/7
T5	5/9	4/9	7/9	6/9	1

	T1	T2	T3	T4	T5
T1	1	0	1	0	0
T2	0	1	0	0	1
T3	1	0	1	0	0
T4	0	0	1	0	1
T5	0	0	1	0	1

Figure 21: term-term similarity matrix **Reduced similarity matrix**

courtesy of : Automatic information organization and retrieval[33]

Some manual adjustment can be made to make it a tree. By the way, coefficient c_{ij} was intentionally made unequal to c_{ji} so that one can verify if term i should be above term j or not. If a concept class grouping is wanted in place of a hierarchy, then a symmetric coefficient function can be used as follows:

$$c_{ij} = \frac{\Sigma_k x_k^i \cdot x_k^j}{\sqrt{\Sigma_k (x_k^i)^2 \cdot \Sigma_k (x_k^j)^2}}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.8 Index Structure Representation:

The information systems is made of two types of files: the data files which contain the information to be retrieved on queries and the auxiliary dictionary files that are used for information analysis. The data files may be simple or complex in structure and so is with the auxiliary file. Now a synonym dictionary may need to contain relations in the form $R_iL_bR_j$ where R_i and R_j are some entities and L_b is a predicate. Now this relational data can be stored in various forms as:a) simple data file of the triplets, b) a graph of nodes representing entities and edges representing predicates, c)using chained links along with the triplets so that all the same predicate values form a linked list, all the same values for the first entities for a linked list and all the same values for the second entities of the triplets form a linked list.

In a stem thesaurus, the entries are made of word stems and word suffixes. A suffix cutoff procedure will generate the word-stems from the text words before the dictionary routine can be applied. Each entry in the dictionary will contain a few semantic codes (concept codes) and some syntactic codes indicating the grammatical categories of the words. Now all the word-stems may be stored in another data file in the form of trees. Every word can be a tree, or going further overlapping parts of two words can be represented by a common sub-tree. In the word tree, Each letter will be a node and the leaf node will have a blank to indicate the end of the word. Thus, two word-stems *add* and *addictive* will appear as in fig 22.

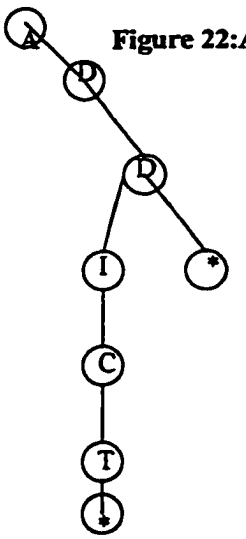


Figure 22:A word tree for word-stems Add, Addict

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In general, a node in such word-trees can have any number of children. To represent such a tree one standard method is substituting all the children pointers by one pointer to the first son and another to the brother as described in fig 23.

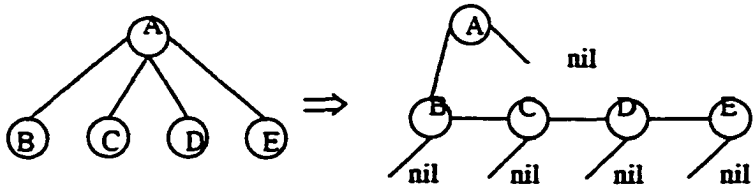


Figure 23:converting an m-ary tree to a binary tree

In general, a thesaurus has two types of operations: thesaurus setup and thesaurus look-up. In the setup operation, an initial loading takes place and subsequently extra additions or deletions are made. In the lookup operation, entries are retrieved from the thesaurus during retrieval operation.

2.9 Statistical Phrase Processing:

The statistical phrase recognition procedure requires that all the phrase components be present within the boundaries of a sentence without any restriction on the position or order of the components. The look-up procedure is made up of taking hold of one dictionary phrase and matching its first component with each component (or stem) of each word of a given sentence and then the second component is matched and so on.

If a particular phrase is found in a sentence, an appropriate entry is made into a chained list of concept numbers, as in a thesaurus look-up. This concept list is sorted by concept no and stored with its weight. A typical entry is shown in fig 24.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A	Concept No	B	Weight

Tag A= 1 for statistical phrase B=1 if the phrase was found in the first sentence of the text.

Figure 24:A typical entry in chained concept list

2.10 Statistical similarity operations:

The concept of calculating similarity between two objects is the notion behind the statistical retrieval procedure in the information systems. Given a set of t terms to characterize documents and search queries, each query can be represented by a t-dimensional vector q in which each component q_i exhibits the value assigned to the ith term. The response in the retrieval system is defined as an assignment of values to all documents where the values reflect the relevance to the query. This defines d-dimensional response vector r for a collection of d documents, in which r_j, the relevance value assigned to document d_j is obtained by comparing the description vector x^j defining the jth document with the query q. The relevance value for the jth document is given by

$$r_j=\Sigma_{k=1}^{t} x_k^j . q_k \qquad (2.1)$$

For a relevance value normalized between 0 and 1, this formula becomes

$$r_j = \frac{\Sigma_{k=1}^{t} x_k^j . q_k}{\Sigma_{k=1}^{t} (x_k^j)^2 + \Sigma_{k=1}^{t} (q_k)^2 - \Sigma_{k=1}^{t} x_k^j . q_k} \qquad (2.2)$$

Now, a document description matrix C of dimension d×t , can be assigned a binary value so that the relevance vector for the d documents are given by

$$\underset{d\times 1}{r} = \underset{d\times t}{C} . \underset{t\times 1}{q} \qquad (2.3)$$

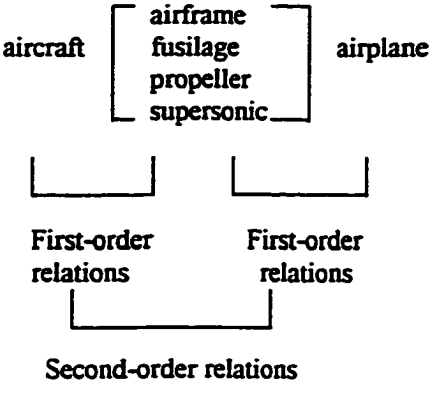
The jth component of this relevance vector r will indicate the relevance of document j varying from 0 to any positive value (depending on the size of the matrix). If the similarity method of eqn 2.2 is to be used, then the formula for the matrix becomes as

$$r = \frac{C . q}{C . C^T + q . q - C . q}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

where C^T is the transpose of C.

The above formulae were built assuming no relations between the document identifiers. But in reality, there are synonymous terms and other statistically related terms. It is assumed that two document identifiers may be related when the same keyword or a synonymous keyword is found to co-occur frequently in the same context. By computing a similarity coefficient between each pair of columns, using the association measure of eqn 2.2, one gets a symmetric term-term similarity matrix $C^T.C$ of dimension t×t and a document-document similarity matrix $C.C^T$ of dimension d×d. These similarity matrices can be reduced to binary form by specifying a threshold value above which a similarity value become 1 and below which the value is 0. These matrix products $C^T.C$ and $C.C^T$ specify first order associations. The second-order associations specified by $(C^T.C)^2$ and $(C.C^T)^2$ relate terms or documents that have common first-order relations. It is conjectured that the odd-powered similarity matrices correspond to contiguity relations and even powers represent synonymy relations. The following example can elaborate the relations:



By taking a weighted sum of powers of the similarity matrices, one can obtain an association reflecting the total effect of associations of every length.

Linear associative retrieval: The simple retrieval equations 2.1 and 2.2 basically reflect keyword mapping. Such a scheme can be replaced by including term and document associations. A possible

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

method consists in assuming that the retrieval transformation is linear (Γ) and that a linear relation exists between term and document vectors. Thus the retrieval vector is given by

$$r = \Gamma . q$$

$$d \quad d\times t \quad t$$

This transformation matrix Γ can be seen as the product of three different linear transformations as

$$\Gamma = A . C . D$$

$$d\times t \quad d\times d \quad d\times t \quad t\times t$$

where A is the d×d document association matrix reflecting associations among documents, C is a d×t description matrix, sometimes called discriminant matrix, containing assignment values to the documents on the values of the terms; and D is a t×t term association matrix specifying known term associations.

It may be noted that the vector multiplication D . q produces a t-dimensional column vector which is something like an extended request vector (vector including the related terms also) , with the effects of known term associations already included. The matrix operation C.D.q produces a d-dimensional vector already in the form of a response vector, including the term-document mapping as well as term-term relationships. Now the product A.C.D.q also includes the documents that are related to the documents of the response vector. So this product A.C.D is an extended transformation of associations. Consider a retrieval system with the following two linear assumptions:

1) The value of a document is assumed to be a linear function of the values of the terms it contains, as well as of the values of the associated documents; i.e. if w is the t-dimensional vector of index term values, then $r = A . r + C . w$ (2.5)

   $d \quad d\times d \quad d \qquad d\times t \quad t$

2) The value of a document identifier, or term, is given by taking the sum of its original value assigned by query q and a linear function of the documents containing it, as well as of the values of the known associated terms,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$w = C^T r + D w + q \qquad (2.6)$$

$$t \quad t\times d \; d \quad t\times t \; t \quad t$$

these two equations eqn 2.5 and 2.6 can be written in a matrix form as:

$$\begin{bmatrix} r \\ w \end{bmatrix} = \begin{bmatrix} A & C \\ B & D \end{bmatrix} \cdot \begin{bmatrix} r \\ w \end{bmatrix} + \begin{bmatrix} 0 \\ q \end{bmatrix}$$

$$d+t \qquad (d+t)\times(d+t) \qquad d+t \qquad d+t$$

where B is C^T.

solving for w, one obtains from eqn (2.5)

$$w = C^{-1} (I - A) r \qquad (2.7)$$

where I is the usual identity matrix with 1s in the diagonal and zeroes in the rest.

Similarly, one obtains from eqn 2.6

$$(I - D)w = Br + q \qquad (2.8)$$

After substituting eqn 2.7 into 2.8 and solution of 2.8 for r, one has finally

$$r = [(I-D)C^{-1}(I-A) - B]^{-1} q \qquad (2.9)$$

By using the identity

$$[(I - D) C^{-1}(I - A) - B]^{-1} = (I-A)^{-1} C (I-D)^{-1}[I - B(I-A)^{-1}C (I-D)^{-1}]^{-1}$$

Eqn 2.9 can be transformed into eqn 2.10, which is in the standard form exhibiting document associations,

term-document mapping and term associations. Specifically,

$$r = (I-A)^{-1} \; C \; (I-D)^{-1}[I-B(I-A)^{-1} C (I-D)^{-1}]^{-1} \; q \qquad (2.10)$$

$$d\times d \quad d\times t \quad t\times t$$

After substituting eqn 2.10 into eqn 2.7, one obtains a corresponding result for w:

$$w = (I-D)^{-1} [I - B(I-A)^{-1}C (I-D)^{-1}]^{-1} q \qquad (2.11)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Now if the term-term and document-document associations are disregarded, i.e. A and D matrices are null,

then eqn 2.10 and 2.11 becomes

$$\underset{d}{r} = \underset{d\times t}{C} \quad \underset{t\times t}{(I - C^T C)^{-1}} \; \underset{t}{q} \qquad (2.12)$$

and $w = (I - C^T C)^{-1} q$ (2.13)

After expansion of the bracketed term in eqn 2.12 one obtains,

$r = Cq\,[1 + C^T C + (C^T C)^2 + \ldots]$ (2.14)

where $(C^T C)^n$ represents nth-order term-term associations derived by matrix multiplication from the term-document matrix. By expanding the series sufficiently far, term associations of any order can be taken into account in the retrieval operation. If all term associations derived from the descriptor matrix C are discarded, then eqn 2.14 results in eqn 2.3.

The first-order term-term associations C^T. C and the corresponding document-document associations C . C^T differ from term or document associations matrices D or A by the fact that the former is derived from the same description matrix C and the others are pre-built document-document and term term-term association matrices. If eqn 2.14 is being used, it is desirable to attach more weight to the low-order associations than to the high-order ones i.e. if L is the term-term similarity matrix, one expects that

$\lim_{n\to\infty} L^n = 0$. To achieve that, the C matrix needs to be normalized with the row sums less than unity. Let λ be a diagonal matrix of dimension (d+t)×(d+t) such that $0 <= \lambda_i^i < 1$ for all i. The desired normalized form is then obtained as:

$$\begin{bmatrix} \delta & 0 \\ 0 & \rho \end{bmatrix} \cdot \begin{bmatrix} A & C \\ B & D \end{bmatrix} = \begin{bmatrix} A' & C' \\ B' & D' \end{bmatrix} \qquad (2.15)$$

(d+t)×(d+t) (d+t)×(d+t) (d+t)×(d+t)

Here δ is a d×d diagonal matrix used to normalize the first d rows of the partition matrix, and ρ is a t×t

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

diagonal matrix which does the same for the last t rows.

$$\delta_i^i = \frac{\lambda_t^i}{\Sigma_{j=1}^{d} A_j^i + \Sigma_{j=d+1}^{d+t} C_j^i} \qquad 1 <= i <= d$$

and

$$\rho_i^i = \frac{\lambda_t^i}{\Sigma_{j=1}^{d} B_j^i + \Sigma_{j=d+1}^{d+t} D_i^i} \qquad d+1 <= i <= d+t$$

The denominator in these expressions is used to reduce each row sum of the partitioned matrix to unity and the constant in the numerator provides a further reduction of this sum to less than 1, thus ensuring convergence of the association matrix.

2.11 Implementation Scheme of Associative retrieval

A retrieval strategy including procedures for manipulating statistically associated concepts are based on at least three different files:

1. A document-term array that specifies for each document the concept name and their weights (this is the C array).
2. An array of concept-concept associations, specifying for each concept the list of associated concepts whose similarity coefficient with that concept lies above a given threshold (D matrix).
3. A tape in document order, containing for each document the bibliographic citations to be used for output purposes (A matrix).

In order to be able to generate the concept-concept associations, it is necessary to store the available document-term array. But the size of such a matrix is so large, that it can not be maintained as an array. In a typical example, a collection size may be 50,000. No of concepts may vary from 5,000 to 20,000 or more. The array size become 1G words. Internal memory size may not fit such a large array. Fortunately, such arrays are largely sparse containing zeroes for most of the concepts. The average no of non-zero values per row will be less than 30 and the rest of the 20,000 entries on each row are zeroes.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Hence it is better to store only the non-zero values. One approach is to store each row in n+2 words where the word 1 contains the current row-no, word 2 contains the no of nonzero items (n), and then n words contain combination of the non-zero association-values (half-word) and the column-no (half-word). Fig 25 gives the schematic description. Now to contain all the values for 50,000 documents, the storage need is 160K words which is reasonable. Now if a new concept needs to be added, lots of data have to moved around to create the extra space. Moreover sorting this data by document no to create a concept-document array will be a hard task. Similarly, There is an updating problem also. If there is a global update on a concept affecting many rows, it takes long time to gain access to the next row.

In a slightly modified scheme (fig 26), a chained strategy is used. Here, the original matrix is stored as two arrays: one pointer array and one data array. The pointer array contains one entry for each row showing pointer to a row in the data array that has the first non-zero value for the given row.

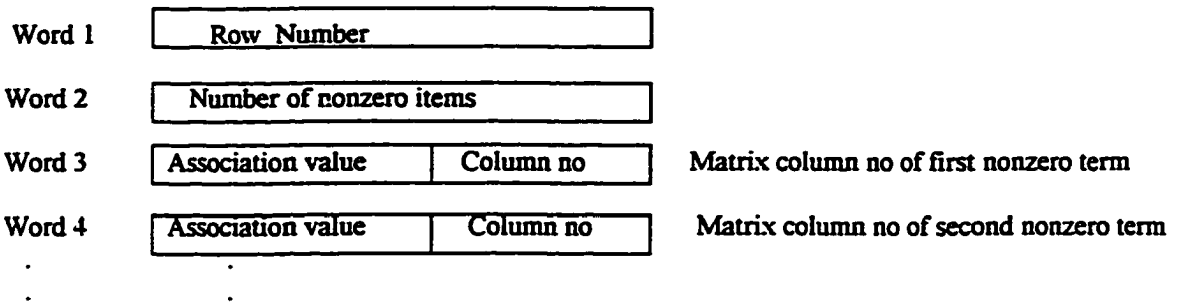


Figure 25:Example document-term matrix

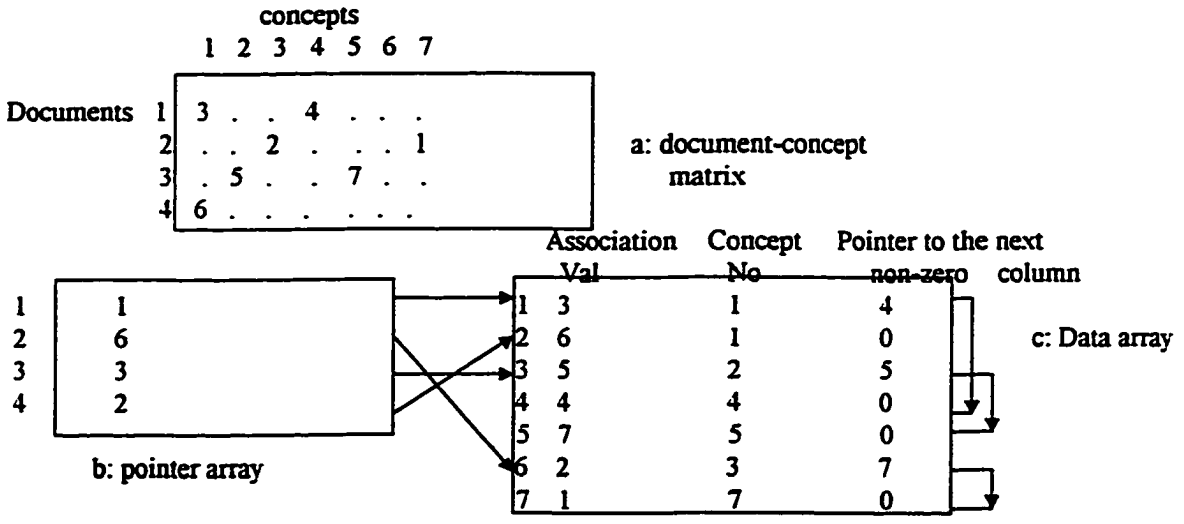


Figure 26:Chained document-concept matrix

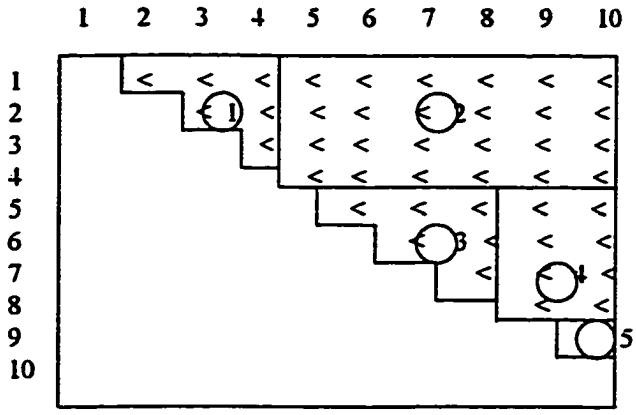
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The data array contains one entry for each non-zero column tied with a chain. So to look up a row, one needs to start from the pointer array and arrive at the proper row no, then follow the pointer to data array to find the first non-zero column and then follow its own chain to wander through all the non-zero columns.

It is easier to update the chained array than the dense map of fig 25, but resorting the array by concept number (for the C^T array) is still difficult to perform. There are systems that make that happen by storing the document no along with the concept no in another word. That means in every entry of the array the concept no, the weight, the document no are repeated.

Now as the document-concept pair can be as large as possible, still the new chained matrix representation may not fit in the memory. So to compute the concept-concept associations (the $(C^TC)^n$ for the nth order association), some part of the matrix needs to be moved from the tape to the memory, carry out the computation, move another part and continue this way. The actual correlation procedure in some system consists in filling a buffer with as many concepts as possible, then correlating each one with the others. Then the remaining vectors from the input tape are passed through one at a time and correlated with all the vectors in the buffer. As these extra vectors are correlated, they are written out to a new tape to be used as the input tape for the next pass. After all vectors on the first input tape are processed, the next buffer load is taken from the new input tape, and the process continues this way until there is no more vector on the input tapes. An example of this is given in fig 27 that calculates the concept-concept correlation of 10 concept-document vectors and assumes a buffer area that can fit in 4 such concept vectors at a time. Vectors 1 to 4 are loaded into the buffer and they are correlated with each other . Then vectors 5 through 10 are taken up one at a time and correlated with the vectors in the buffer. The next buffer load consists of 5 to 8 and then the final one handles 9 to 10. Since the correlation coefficient matrix is a symmetric one, only n(n-1)/2 coefficients are important here.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Step 1. Fill buffer with vectors 1,2,3,4; perform correlation among each other.

Step 2. Enter one vector at 1 time and correlate with buffer; write out input vectors onto a new tape (i.e. vectors 5 through 10).

Step 3. From tape produced in step 2, enter a new buffer load (vectors 5,6,7,8), and correlate each with each.

Step 4. Repeat step 2 with remaining vectors (9,10).

Step 5. Enter a new buffer load (vectors 9 and 10) and correlate.

Figure 27:Example concept-concept correlation process

The output format of the correlation process consists of two-word items where word1 contains the concept pair (concept A and concept B) and word2 contains the coefficient calculated.

Concept A	Concept B	Association value

Now the association results should be converted into binary values by comparing the coefficients with a threshold chosen judiciously (the greater and equals get 1 and others 0). Finally, these values should be sorted by concept A and then concept B. Now repeating the process one more time, the second-order correlation matrix $(C^T C)^2$ can be produced and so on. At a point, the expanded concept-concept correlation matrix is computed sufficiently precisely. The concept-concept matrix can be used to expand the document matrix or the query matrix by including the concepts that are related to given concepts. A flowchart of the overall concept association process is given in fig 28.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

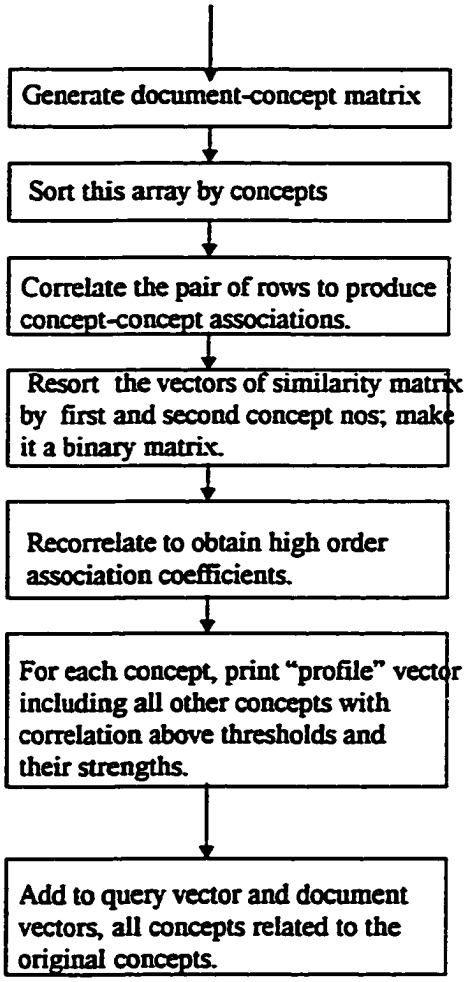


Figure 28:Flowchart for concept association

The standard concept vectors that result from the thesaurus lookup are transformed into two word formats as described before. Then the document-concept matrix is transposed (or sorted) to create concept-document matrix (from d×t to t×d matrix). Then the concept-concept correlation is created using the steps in fig 27 and the result matrix is resorted by concept numbers. Either it is converted to binary results or values less than a threshold value are replaced as zero (to take care of spurious noises). The association process is repeated if higher order coefficients are expected. Finally the document and query concept vectors are expanded to take into account these related concepts.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The same type of idea can be used to compute document-document correlation just like the concept-concept correlation. This will produce documents related to each document. Such matrix may be used to expand the answers for the queries to include additional documents which are related to the result documents as per the document-document correlation matrix. It will be easier to calculate the document-document correlation matrix, because the document-vectors are already ordered by documents and higher order coefficients are not necessary for the calculation.

2.12 Experimental results for concept correlation

All the theoretical rigorous models of concept associations have not been practically applied to a large degree (except for some experiments) because there is matrix inversion involved and also the large sized matrices are hard to manipulate. The simpler version of the mathematical model (where concept-concept and document-document associations are ignored) have been used where a few first terms of $(C^T.C)^n$ series polynomial have been used. Various alternative procedures exist to deal with different requirements. One, associations of varying depth are normally put in place (some use first-order, some use second-order). Two, various entities can be used for the associations including words, word-stems, concepts used from thesaurus look-up etc. Three, a large number of parameters can be adjusted leading to different results. Two main parameters are varied, one, the cutoff value for the correlation coefficient (K) below which a statistical association is not recognized and the frequency of occurrence of the terms that are correlated. If all the terms including ones with low-frequency of occurrence are correlated, noisy correlation may be found. This result is reflected in the data shown in table in fig 29 that proves that frequency restrictions imposed on the concepts before association influence the number and type of associations calculated. For example, 83 concept-concept pairs are generated when all concepts occurring at least four times in a given document collection are used, but only 39 such pairs come out when no of occurrences is made stricter to at least 6 for a concept. Also, the proportion of the semantically acceptable pairs (i.e. pairs found in Roget's thesaurus) is larger with stringent restrictions.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The noisy associations show up more as the frequency restrictions are relaxed, thus if one allows all concept-pairs including the ones at low frequency of occurrences, the quality of association matrix degrades.

Example collection

Total no of documents:	400	Total no of text words:	39,000
Tot # of thesaurus concepts:	2,526	Tot no of concept-concept pairs above cutoff of 0.60	10,548

No of occurrences of concepts	No of concept-concept pairs above cutoff	No of pairs with thesaurus relation	Percentage of correct pairs
4	83	14	17
5	52	10	19
6	39	13	33

Figure 29:Frequency characteristics of concept-concept correlation

Although, generally it has been noticed that word-word correlation improve basic word-stem matching process for high recall values, still it has been found that a well-constructed thesaurus is generally more powerful than the associative techniques applied to words.

2.13 Document clustering

The basic classification problem lies in creating some categories for a set of items by assigning the items predictably to a small number of subgroups so that the members in each subgroup are alike. It becomes a problem of optimization of best balancing the loss of accuracy (by representing the individual by an average) and gain in economy. The objective here is to generate some classes from the concepts (that represent the documents) thus leading to document clusters. These clusters really speed up the query matching process and simplify the retrieval of relevant documents. If the relation between the concept groups is a generic one, then the classes are hierarchical else it becomes a 2-dimensional network-based. There are various techniques available for classification, for example, matrix eigenvalue analysis, factor analysis, latent class analysis, clump theory, dense clustering.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.13.1 *eigenvalue and factor analysis:*

The eigenvalue analysis for document clustering is based the properties of reducible matrices. where the set of documents can be partitioned into non-overlapping subsets in such a way that within each subset a non-zero connection exists between each pair of documents, but no relation exists to any member outside the subset. The case may be relaxed by permitting weak connections across clusters.

First consider, a document-document or concept-concept similarity matrix which was discussed before. Such a n×n square matrix A can be called reducible if through row and column-permutation the matrix can be reduced to a form:

$$\begin{bmatrix} X & Y \\ 0 & Z \end{bmatrix}$$

Where 0 is a null matrix of all zeroes and X and Z are square matrices. Maintaining the correspondence between the matrix and a directed graph, an element a (i, j) for row i and column j will indicate the weight for an edge from node i to node j. Now matrix A is reducible if and only if the corresponding directed graph is not strongly connected (i.e., there node p and q in the directed graph such that no path leads from p to q).

When reducible matrix A is symmetric as most similarity matrices are, reducible matrix becomes a block-diagonal matrix which means there is a permutation matrix P that will transform the matrix into a block-diagonal form as

$$B = PAP^T = \begin{bmatrix} X & 0 \\ 0 & Y \end{bmatrix} \begin{matrix} r \\ n-r \end{matrix}$$
$$\qquad\qquad\quad\; r \quad n-r$$

where X and Y are square matrices.

An example of the permuted form of a symmetric reducible matrix is given in fig 30. The corresponding directed graph has disconnected nodes as can be seen in the example. It is clear that the eigenvectors of B has two classes: one class with last n-r components equal to 0 and the other class with first r components

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

equal to zero. These two classes are mutually exclusive and can be termed disjoint clusters. If the further assumption is made that all elements of A are non-negative and that A has been completely reduced to the form:

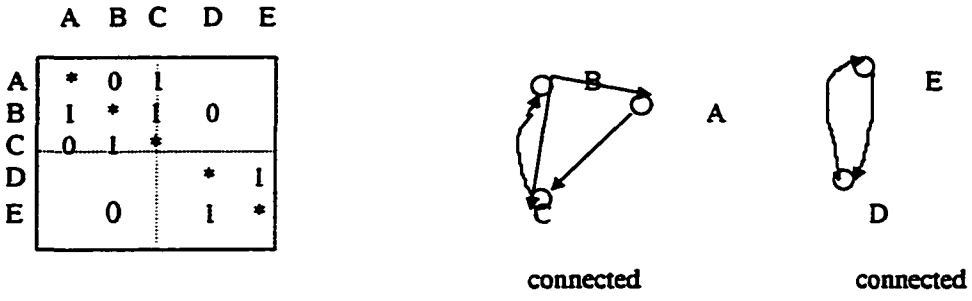


Figure 30:reducible symmetric matrix **(b) corresponding directed graph**

$$B = PAP^T = \begin{bmatrix} X & 0 & 0 \\ 0 & Y & 0 \\ 0 & 0 & Z \end{bmatrix}$$

where each diagonal block is irreducible, then the classical theory (Perron & Frobenius) applied to the matrices X,Y etc. yields the result that each of the irreducible matrices along the main diagonal has a positive maximum eigenvalue to which corresponds an eigenvector, all of whose components are nonnegative. The other eigenvalues are all strictly smaller and their corresponding eigenvectors do not possess the property of nonnegativity.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 2.1 results of the eigenvalue analysis

calculated on a 13 × 13 similarity symmetric matrix

Eigenvalues	5.067	1.801	1.445
Eigenvector indices	First eigenvector	Second eigenvector	Third eigenvector
1	0.271	0.075	0.412
2	0.172	-0.014	0.387
3	0.201	-0.070	0.491
4	0.232	-0.083	0.357
5	0.344	-0.176	-0.214
6	0.337	-0.271	-0.164
7	0.340	-0.254	-0.214
8	0.333	-0.110	-0.052
9	0.331	-0.300	-0.213
10	0.205	0.449	-0.311
11	0.248	0.363	-0.169
12	0.209	0.517	-0.025
13	0.295	0.334	0.136

Courtesy of Gerard Salton: "Automatic Information Organization and Retrieval" [33]

This means, if the eigenvalues of the entire matrix are obtained one at a time in a descending order of magnitude, the largest eigenvalue will be the maximum eigenvalue of one of the sub-matrices. The corresponding eigenvector has positive elements in the positions corresponding to the members of the index set identifying the sub-matrix and can therefore be used to identify the cluster specified by this sub-matrix. The next largest eigenvalue may belong to the same sub-matrix, in which case the corresponding eigenvector

may have negative components or may select the same cluster again, but it may be possible that the second eigenvalue is the highest root for another sub-matrices; in that case the eigenvector selects the elements of the second cluster. This way, the eigenvectors corresponding to successive eigenvalues may come from the different clusters present in the original matrix. For example, table 2.1 displays three clusters obtained by identifying the three largest eigenvalues of a 13×13 similarity matrix. The clusters include documents as 5, 6, 7, 8, 9 in cluster 1; documents 10, 11, 12, 13 in cluster 2; documents 1, 2, 3, 4 in cluster 3.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Thus in the example, when the similarity matrix is reducible to block-diagonal form, any standard method to calculate the eigenvalues and the corresponding eigenvectors of the matrix can be used to find the clusters in the that matrix. At times some similarity matrices cannot be transformed into such a block diagonal form so easily through row and column permutations and the eigenvectors do not show very clear distinction between large positive and negative values. The eigenvalue analysis produces clear results. Anyway, the eigenvalue analysis can be extended from perfect clustering to strong clustering, weak associations are allowed between members of different clusters. In this situation, the zero elements in the reduced matrix are replaced by small values ε. The properties of such ε-reducible matrices can be deducible from the perfectly reducible matrices. Mainly, it can be shown that to first order in ε, the elements of the eigenvector are changed, if by an order of ε and the previously described structure of the eigenvectors are preserved. Thus the eigenvalue analysis is extensible to the situation of ε-reducibility.

When the clustering becomes weaker and the ε-elements in the permuted similarity matrices increase further in values, the difficulty in finding large components of eigenvectors becomes clearer. This consistently reflects the problem that arises in deciding cluster members irrespective of the method for eigenvalues that is used (proving the clustering is weak).

The factor analysis method may be considered as more general method than eigenvalue analysis method differing in complexity of the calculation the type of original similarity matrix on which the calculations start. The method factor analysis can be used to calculate the clusters if and only if :

1. The pair-wise similarities are expressed by the standard product-moment correlation.
2. The factors that are determined turn out to have the properties assumed for the eigenvectors of "almost separable" matrices.

The standard product-moment correlation for two n-dimensional vectors X and Y is given by:

$$r(X,Y) = \frac{\sum_{i=1}^{n} X_i . Y_i - 1/n \sum_{i=1}^{n} X_i . \sum_{i=1}^{n} Y_i}{\{[\sum_{i=1}^{n} X_i . X_i - 1/n .(\sum_{i=1}^{n} X_i)^2][\sum_{i=1}^{n} Y_i . Y_i - 1/n (\sum_{i=1}^{n} Y_i)^2]\}^{1/2}} \qquad (2.16)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A few tests were done where subject categories were drawn by factor analysis and then compared to manual classification schemes. Some agreements were found and sometimes unique classification could not be done.

2.14 Overlapping clustering

Although the above methods lead to non-overlapping clusters, but they involve matrix multiplications and inversions. There are other simpler methods that do not impose so restrictive conditions for membership, but result in overlapping clusters i.e. the same document belongs to many clusters. Such overlapping clusters certainly can be used for indexing where the cluster attribute for a document can be used as an index as word-stems, thesaurus concepts etc. Overlapping clusters are also useful during document-request matching operations which happens in the first step of retrieval operations.

One such method for this uses the term-document matrix as input, which is later transformed into a term-term or document-document similarity matrix as described before. The similarity matrix is then converted into a binary matrix by choosing a threshold similarity and then using it for comparison. This operation can be repeated many times thus generating second-order, third-order associations. Clusters are those sets of documents of maximum size where the documents are similar to each other (each document is similar to all other members). As this method may generate a large number of small, possibly non-overlapping clusters, an attempt is made to enlarge the clusters by dropping the need for a strong association with all other members in the cluster. This is done by transferring a member of a cluster with membership less than a predetermined constant into the large cluster to which it is most similar. This similarity is the percentage of items in the large cluster to which the transferring document is linked in the first-order item-item similarity matrix. Before accomplishing the transfer, this percentage is checked against a threshold, below which a transfer is not done.

After all the transfers are made, a *classification vector* is generated for each cluster by taking the mean value of the vector elements of all the items within the cluster to define the corresponding vector for the class. This classification vector can then be used to represent the cluster during query retrieval

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

procedure. Another clustering method creates cluster vectors to be used efficiently for search requests. From the results of the previous search requests, such vectors (that is close to search requests) are created that maximize the search efficiency and are akin to the previous search results.

There is another method due to Rocchio which is claimed to be parameter driven. Here number of clusters to be generated, the cluster size and the amount of overlap between the clusters all can be controlled. The method expects the following input parameters:

1. The number of clusters needed.
2. Lower and upper bounds on the number of elements to be included in a cluster.
3. Lower threshold on the association between an item and a cluster classification vector below which the item can not be classified in a cluster.

Initially all the items are considered unclustered, and they pass from this state to one of the two possible states, clustered or loose. An unclustered item is considered as a possible candidate for a cluster center. All the other unclustered and loose documents are correlated with it and the candidate is subjected to a region density test to see if a clustered should be formed around it. This test requires that more than n_1 items should have correlations higher than threshold p_1 with the candidate and more than n_2 items should have correlations higher than threshold p_2 with the candidate. This ensures that items on the edge of a large group does not get selected as a center and also an annular region where many items are concentrated in a ring-like area around the candidate item are not accepted as clusters. Fig 31 shows some cases of test failures for cluster formation. The radii p_1 and p_2 are indicating the minimum threshold strengths of the associations for the inner circle and the outer circle (the proximity indicates the closeness or distance of associations). The space is the concept vector space and the points are the documents. In fig 31a, the test failed because less than n_2 documents are within distance p_2 (the outer circle is too thin) although the first condition for distance p1 is good. In fig 31b, the test failed because less than n_1 documents are within distance p_1 (the inner circle is too thin). In fig 31c, the test failed because very few documents (rather less than n_1 documents) are within distance p_1 and also very few documents are within distance p_2.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

a) less than n_2 items within distance p_2 b) less than n_1 items within distance p_1 c) less than n_1 items within distance p_1 and less than n_2 items within distance p_2

Figure 31:Failure of density test

If the density test fails for a given item, the item is considered "loose" and another unclustered document is chosen. If, the item passes the density test, a cutoff correlation p_{min} is determined as a function of the category size limits and the distribution of the correlation values. Documents with a correlation above p_{min} are automatically placed above cutoff; if the correlations fall below p_{min} before the size limit is exceeded, the cutoff is chosen at the greatest correlation difference between adjacent items, for creating the sharpest boundary between identified subset and neighboring unclustered items. It is still possible to have some items included in many adjacent groups.

Given a set of D documents included in a cluster, the centroid vector (c) is computed to represent the cluster from D.

$c = 1/n \Sigma_{d \in D} d$ where d is the document vector within the cluster and n is the size of the cluster. The cutoff parameters on category size is reapplied to create an altered cluster. Three results are possible for each item:

1. The correlation coefficient of the item may exceed the cutoff value but may at the same time lie below p_{min} ;such an item is marked "loose" to prevent its choice as a possible cluster center.
2. Items with correlation below the cutoff value are considered unclustered.
3. Items with a correlation value above the minimum p_{min} are considered clustered.

The situations that may happen are case 1)where the correlations fall below p_{min} before the cluster size limits are reached. And case 2) where the size limits are reached before the correlations fall below p_{min}. In

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

case 1, the computed cutoff will be in between p_{min} and the size limitations thus giving rise to documents that are clustered, loose and unclustered; in the latter case, documents are either clustered or unclustered.

This process continues with all the unclustered items, and the first pass ends when all items are either clustered or loose. Since the classification vectors are correlated with the entire collection, some items may show up in many different clusters.

A second pass may be used if the number of categories is less than the required value. This is done by using as cluster centers some items that originally failed the density test. If too many classes came up in the first pass, the density test could be made more restrictive, or the class size limits could be increased.

Some relatively isolated items may still be not clustered at the end of the second pass. Several strategies can be used for these items: They can be assigned to the closest cluster, they can be considered as singleton clusters by themselves, or a third pass can be used to cluster those. Such a third pass might be the formation of a partition class for classification vector for each set within the partition, and a new clustering operation around the centroids.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

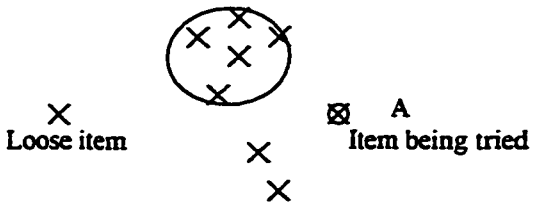
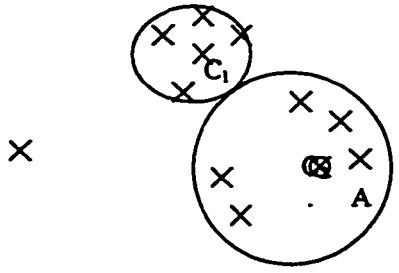
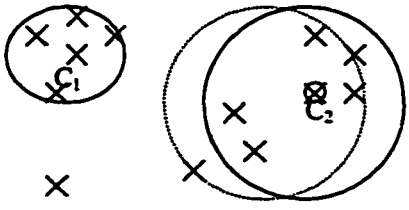
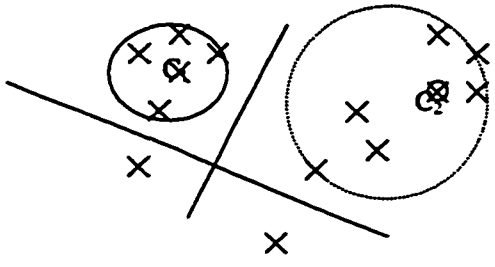
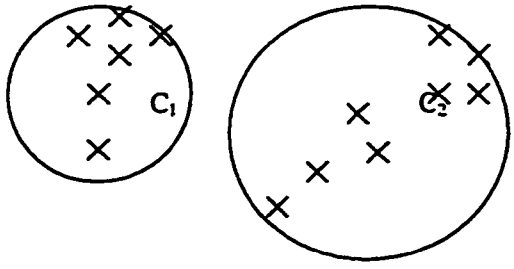
Steps	Drawing
step 1 Pick up an unclustered item and match all the unclustered items with it.	
step 2 Submit document A to density test; if test fails, declare A to be loose; if test works, items are clustered around A to compute a centroid vector C_2 for the new cluster.	
step 3 Take centroid C_2 and match it in conjunction with all documents including clustered, unclustered and loose; redefine a new cluster around C_2, which again leaves items either clustered, or unclustered or loose.	
Step 4 To reduce overlap between clusters, form partition of documents around centroids; recompute centroid vectors and recluster.	
Step 5 Add loose documents to closest cluster.	

Figure 32:Depiction of Rocchio's clustering process

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

To summarize, the complete process includes three clustering operations: the first around the initial items that pass the density test; the second around the centroids of the clusters previously generated; and the third around the new centroids obtained after partition of the previous sets. Fig 32 explains the steps graphically.

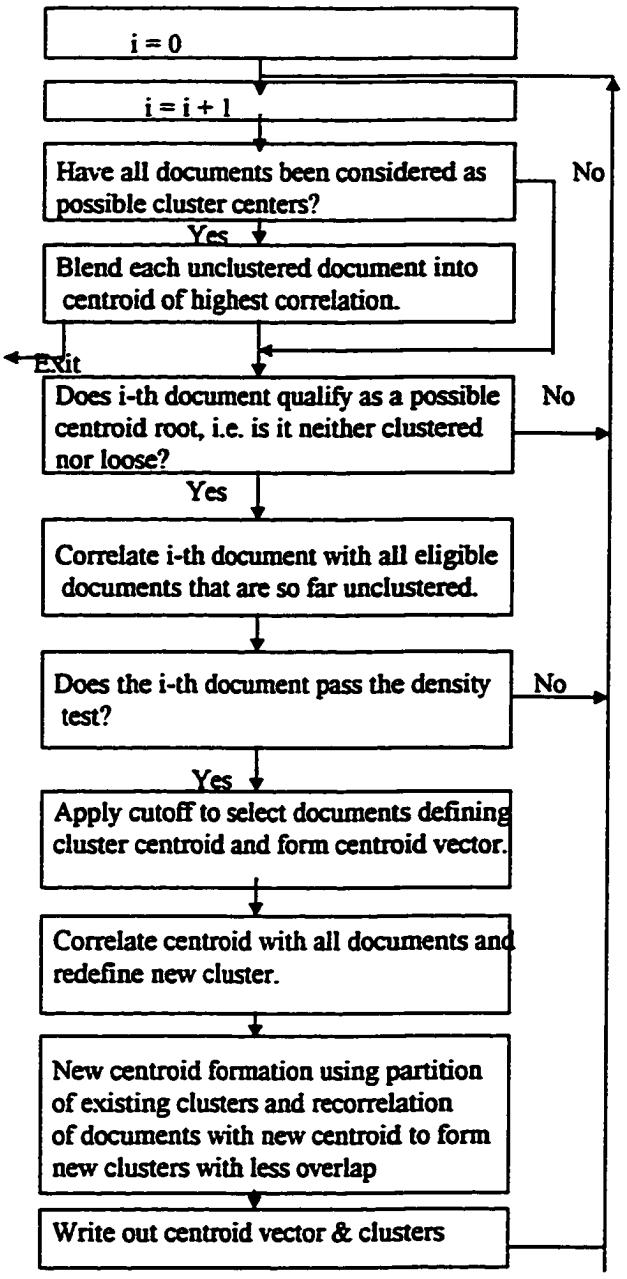


Figure 33:Simple clustering method

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.15 Term-sentence clustering:

Another weak clustering procedure commonly found, is term clustering based on co-occurrences of terms within the sentences of a document or a document collection. This process is based on an initial partition of the set of objects to be clustered. followed by the transfer of items from one group to another group to maximize the connection strength within each group and simultaneously to minimize connections between groups.

Consider the set U of all n concepts to be clustered and A be a subset of U with elements a(i). i=1,2,...t; let B be the complement of A, with elements b(i), i=1,2,....,r where n = r + t. If x and y are objects

in U, then let c(x,y) represent the association between objects x and y, as specified by the corresponding element of the association matrix S .

The following quantities may now be defined:

c(x, y) is the association between objects x and y, where c(x, x) is assumed equal to zero for all x.

$S(x, A) = \Sigma_{i=1}^{t} c(x, a(i))$ is the total association of any object x to all objects in A.

$S(x, B) = \Sigma_{i=1}^{r} c(x, b(i))$ is the total association of object x to all objects in B.

b(x, A) = S(x, A) - S(x, B) is the bias of concept x to set A (positive or negative).

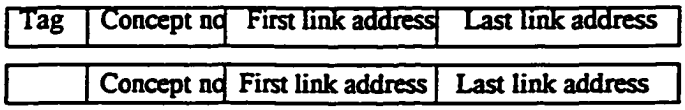
A cluster may be defined as the set of all x that have positive bias to A so that, each element that is outside A have negative bias to A; more specifically,

$A = \{x: b(x, A) >= 0, \text{ and } b(y, A) < 0 \text{ for all } y \in B\}$

One clustering method is made up of choosing one of the objects from U, say object u_j, and defining an initial set A_j containing the object u_j as well as all other objects that have a non-zero association with u_j. Next, the bias $b(x, A_j)$ is computed for all objects x in U, and objects with positive bias are transferred to A_j if not already there. After each transfer all biases are recomputed, and the iteration continues until no further transfer takes place. The final A_j obtained is then defined as a cluster and then the same procedure is repeated for the next object in U after u_j.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

This procedure is exhaustive and so a little tedious. The efficiency depends on how accurately the initial sets A_j are chosen. Another problem is that, the cluster sets may become too large for practical purposes. In such cases, some elements have to removed. A bias threshold value is chosen (say b_1), and then the element in the cluster with lowest bias and bias value less than b_1 will be eliminated out. At this time, again all the biases of the members of U are computed and more transfers are made. Finally the reduced cluster has at least one (or more) less element. This method is due to Needham and is flexible to accept parameters as maximum cluster size etc. This method is used to build clusters in secondary indexes for the documents. For example, the 10-word items produced through dictionary-lookup, representing the word-stems included in a given document together with the corresponding concepts are first converted into a chained array containing the word concept, sentences containing it and the weights. There is a chained array for each concept (in fig 34b). There is an array of concept elements in a table that contain anchor for the chained array (as in fig 34a).



(a) entry in concept no table

(b) chained list of concept occurrences

Figure 34:An example concept-sentence matrix

All the occurrences of the same concept in the same sentence gets added as weight in the chained list. Now these structure can be translated into three-word table where each entry contains concept no, weight and sentence no. Now these concept-sentence matrix association can be calculated to form concept-concept association and the concept-concept associations can be used to compute concept clusters and the cluster identifiers.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

If the document collection is homogenous, the same concept clusters may be generated for many documents and the clustering method can be used to replace any manual thesaurus. In general, the document collections may not be so homogenous (in terms of percentage of statistical occurrences) to result in unambiguous clusters. In such cases the correlation between pairs of concepts can still be used for document identification.

2.16 Syntactic Operations

Although the terms generated by dictionary look-up identify the main topic classes (using the methods above)in the documents, still the phrase identifiers from the documents are very useful in specifying some of the principal ideas contained. However these phrases are mostly isolated from the context and hence the relationships between the main ideas and these phrases mostly remain unstated. It has been shown that the dictionary look-up and phrase-indexing procedures (which have been discussed in prior chapters) are very important in an automatic retrieval system. The details of the syntactic analysis will be described below.

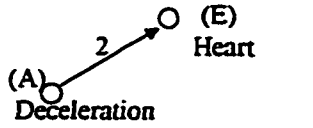
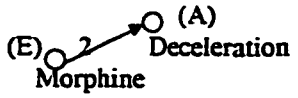
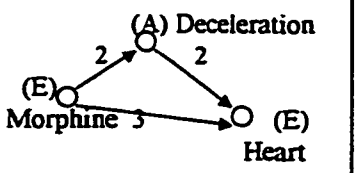
2.16.1 *Automatic content analysis:*

Most information systems today look for important keywords in the document and combine these keywords into larger units for indexing the documents. Although, the identification of the keywords from the sentences is easy, but forming an interpretation of meaning of the words is very difficult today. There are no well-defined rules that can be used to combine the keywords into a group. There are sometimes no way of resolving some of the syntactic and semantic ambiguities with the words. But the correct meaning of the word group largely depends on resolution of ambiguities, interpretation of homographs, recognition of equivalent semantics, a general knowledge of the background and environment. To deal with great complexities of natural languages, it is suggested to simplify the complexity further and thus limit types of relations among keywords. Two simple classes may be permitted : *analytic* and *synthetic* relations. The *analytic* relations include generic or inclusion relations pointing to a hierarchical classification. The second one is a real dependency relation (e.g. cause-effect relation between poison and death - two keywords) between keywords.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

To perform an effective content analysis of a text, it is necessary to replace the keywords by thesaurus concepts and then arrange these concepts by their analytic and synthetic relations. Then a text can be represented by a tree of the analytic relations among concepts or in general a graph of concepts and their relations. Construction of such an information graph is better explained using an example in fig 35. Now the clause that has been used :"*Morphine acts as a decelerator of the heart.*" is very simple one. Analyzing carefully the phrase, "*decelerator of the heart*" one finds an associative relation between word-stem "*decelerate*" and "*heart*". Also there is another associative relation between *Morphine* and the word-stem *decelerate* which is of the type cause-effect. On the whole there is a consecutive relation between words *Morphine* and *heart*. As a small, syntactically simple sentence establishes so many relations, one can imagine how big the graph becomes when the sentence is complex. If a large paragraph is used for such an analysis, the semantic graph becomes really very complex. However, the basic unit of a relation is only a few words (2 or 3) in a phrase, e.g. a noun phrase, a prepositional phrase, adverbial phrase or a simple subject-verb-object grouping. Such phrases can be also be identified by syntactic analysis of documents. That is why syntactic analysis is so crucial in document content analysis where it adds additional principal ideas to the main concepts.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Graph form	English text
	Deceleration of the heart (part)
	Morphine acts as decelerator (part)
	Morphine acts as decelerator of the heart (full)

2 Associative relation
3 Consecutive relation
A Action
E Entity
S State

Figure 35:Simple semantic graph for phrase showing synthetic relation

2.16.2 *Syntax and Semantics*

Phrase structure grammars: For automatic syntactic analysis, a grammar needs to be specified that will correspond to the text. The complexity of the grammar can be high or low, but a typical is phrase structure grammar in which a sentence is considered to be a juxtaposition of nested phrases. Phrase structure grammars for a language are specified by some rewriting rules as:

$\alpha A\beta \rightarrow \alpha\omega\beta$ where A is grammar symbol and α, β and γ are strings of characters in the language.

When α and β are null, the rewriting rules leads to a context-free grammar where non-terminal A can be replaced by a string ω irrespective of context; otherwise it is a context-sensitive grammar. Given a grammar and a sentence producible by the grammar, it is possible to show how to derive the sentence using the rewrite rules and this is called the *derivation*. The derivation of a sentence is a grammar can be shown by a tree called a *phrase marker*. During document analysis the problem is that of recognition, i.e.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

arriving at a phrase marker for a sentence for which a derivation exists according to the grammar.

Suppose, there is a grammar as follows in fig 36.

1. <S> → <NP><VP>
2. <NP> → <PP>
3. <VP> → <V><NP>
4. <NP> → <A><N>
5. <PP> → Those
6. <V> → are
7. <A> → fleeting
8. <N> → saucers

S: Sentence
NP: Noun phrase
VP: Verb Phrase
PP: Personal Pronoun
V: Verb
A: Adjective
N: Noun

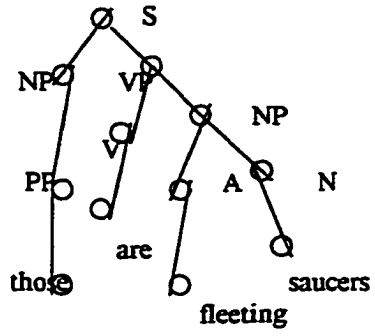
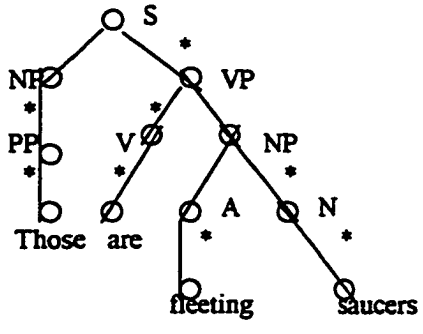


Figure 36:Sentence derivation

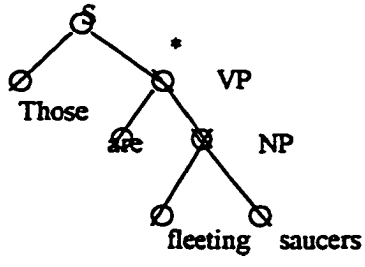
Context-free grammar Sentence: "*Those are fleeting saucers*" (b) sentence derivation

It is clear that the three phrases are identified by the phrase marker of fig 36(b), made of

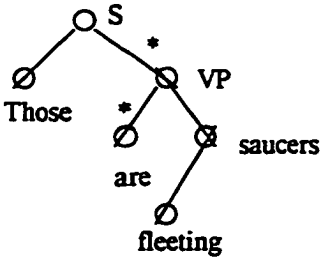
Noun Phrase : (fleeting saucers)
Verb Phrase: (are (fleeting saucers))
Sentence: (Those(are(fleeting saucers)))



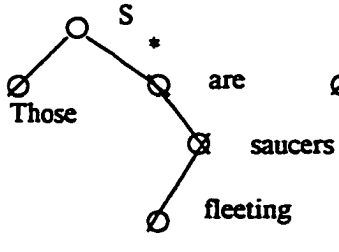
(a) Initial phrase marker (* for a governing node)



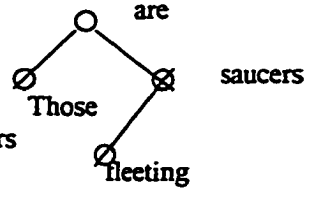
(b) Phrase marker after erasing branches with degree 1



(c) After processing ((fleeting) saucers)



(d) After processing (are ((fleeting) saucers))



(e) final dependency tree ((Those) are ((fleeting) saucers))

Figure 37:Converting a phrase marker into a dependency tree

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

There are other types of derivation than phrase marker. One of it is *dependency tree* in which there is a distinction between the two minimum elements of a phrase, one element of which is governor and the other is dependent. In a dependency tree, only the terminal elements of the phrase markers are included and in a syntactic dependency tree, the dependent node sits below the governing node. To convert a standard phrase marker into a dependency tree, one has to identify all the governing nodes and mark with an asterisk by the side of the branches leading to the governing node. Whenever the node corresponding to a word is joined to its superior node, the inferior node and the connecting line are erased, and any dependents of the inferior node are transferred to the superior node; this process is continued until no lines marked with asterisk remain.

2.16.3 Phrase structure parsing

A number of parsing methods are available for parsing context-free phrase structure grammar, one of which is predictive analyzer. This tool operates by using a set of "predictions" of possible syntactic structures to be found during parsing. These predictions are stored in a stack called "prediction pool". Analysis of a given sentence continues from left to right, a "sentence" prediction is initially pushed into "prediction pool" stack. The input string is a sequence of homograph sets, one set for each word in the sentence. Each homograph determines the possible membership of the corresponding text word in a syntactic word class and the homograph sets are originally obtained for all words in each sentence to be analyzed by a dictionary look-up procedure. For example, the word base will have one homograph assignment as *noun, singular*, one for *transitive verb*, and one for *adjective*.

To start with, the "sentence" prediction is combined with the first homograph of the first word in the sentence; the resulting argument is looked up in a *grammar table*. If the syntactic word class for this homograph can begin a sentence, one or more sets of new predictions, called subrules are pushed into the stack after popping the stack. On the other hand, if no entry that pairs the current prediction and input homograph matched the grammar table, that path is abandoned and the second homograph for the same input word (first word) is chosen and a pair is matched in the grammar table. Eventually, all homographs

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

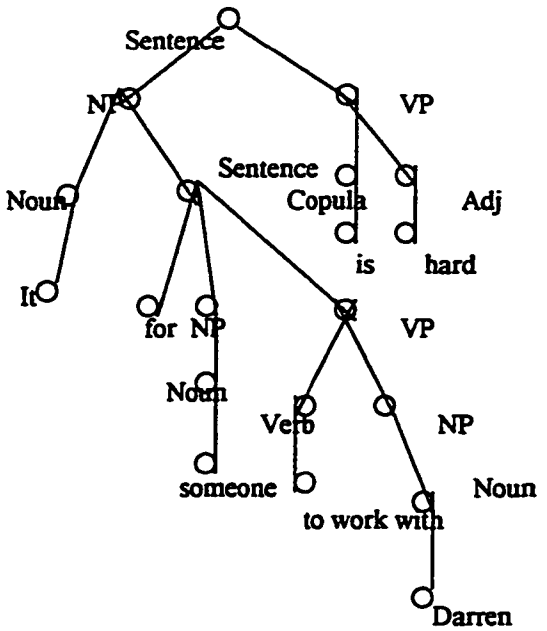
are tried at each word position, and each set of new predictions obtained from the grammar by a particular argument pair is entered into the prediction pool at least once. This multiple-path technique ensures that all possible paths will be tried. If no derivation path exists, then the analysis fails. If the input sentence is well formed, and a corresponding phrase marker is produced (as the stack is popped and pushed), after the last word has been processed the stack must have been empty. The predictive parser proceeds in a single left-to- right scan of the input sentence, using rules that specify the sentence structure upto the end from the point at which the rule applies. Since rules can be embedded in other rules, the number of analyses produced and the time taken to produce them increase fast with sentence length.

Another grammatical method, implemented by a left-to-right scan of each sentence, is called *string analysis*. During parsing, the read sentence is decomposed into a few elementary strings and an attempt is made to decompose the sentence into a few "center strings" (like a skeleton sentence), together with a few modifying adjunct strings ("in general", "on any event", "thank goodness") and replacement strings to replace some parts of other strings. (for example, "Mr. Ranger" can replace "that stranger" in the sentence "that stranger looked aghast."). This string analysis grammar includes some "discontinuous constituents" which can be accommodated by superimposing on a context-free grammar certain context-sensitive rules. However a different grammatical theory is needed to handle some of the standard linguistic transformations.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

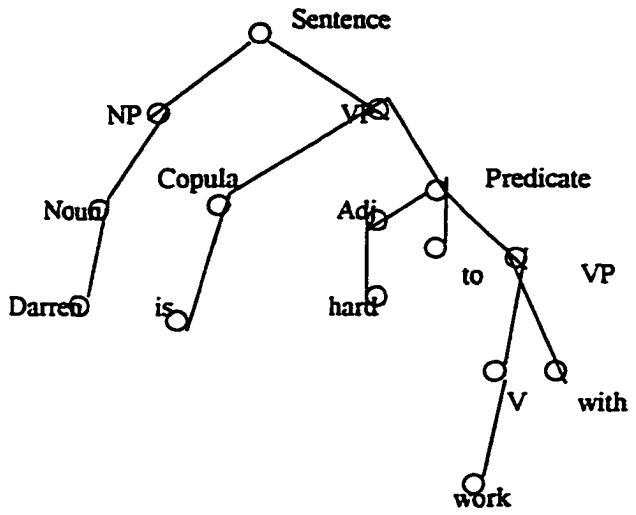
2.16.4 Transformational Grammar

A transformational grammar is superior to a phrase structure grammar due to its semantics handling capability because assigns two structures upon parsing a sentence, called the deep structure and the surface structure. The deep structure makes a semantic interpretation of the sentence whereas the surface structure accounts for the grammatical syntactic analysis.



(a) sample deep structure

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



(b) sample surface structure

Figure 38:Semantics and syntax for a given sentence

A sample deep structure for the sentence " Darren is hard to work with" is shown in fig 38 below. It is visible, that Darren is the object of the verb *work with* and the subject of the verb a unspecified person designated as *someone*. Furthermore, the subject of *is easy* is not *Darren* but the noun phrase given on the left-hand side of the tree structure. On the other hand, Darren is the subject of *is easy* in the surface structure, a fact that can not contribute to the proper semantic interpretation of the sentence. A close look at fig 38b shows that phrase structure grammar assigns only surface structure of a sentence.

The transformational grammar consists of two components, known as base component and transformational component. The base component generates for each sentence a set of terminal strings with an associated structural description of the type shown in fig 38a and known as *generalized phrase markers.* The transformational component maps phrase markers into other phrase markers by a sequence of simple transformations including substitutions, deletions and adjunctions. For example, the transformational component can be used to turn a declarative sentence into an interrogative one and more generally to accept as input a generalized phrase marker and to produce a derived phrase marker.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Transformational rules are applied in an ordered sequence, so that some types of transformation are applied before some types .

As an example, a passive transformation is applied before a number agreement transformation, since the number of the predicate verb in the passive voice agrees with the noun phrase before the verb rather than with the noun phrase following the preposition *by*. The detailed steps involved in an active-passive translation are in given in table 2.2.

Table 2.2 Steps in active-passive translation

Operation	Sample sentence
Initial	The bunnies ate the bananas.
Phrase marker	The bunnies (past) eat the bananas.
Passive transform	The bananas (past) be (en) eat by the bunnies.
Number agreement transform	The bananas (past plural) be (en) eat by the bunnies.
Final	The bananas were eaten by the bunnies.

The recognition procedure for a transformational grammar is more complex than one for phrase structure grammar, because the grammar can not be expressed simply by some rewriting rules. Since a transformational grammar is capable of mapping a deep structure into a surface structure, most existing procedures try to reverse-transform a derived phrase structure into a generalized phrase structure and then compare that with the given generalized phrase structure. If it matches, then the procedure identifies the deep structure as correct semantic interpretation of a given sentence for which the given generalized phrase structure can be transformed into the same structure as the derived phrase structure created by the phrase structure grammar.

2.16.5 *Semantics*

As it may seem from prior discussion that today no automatic syntactic analysis system is available for generating the semantic interpretation for well-formed English sentences. At best, certain

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

key phrases can be isolated from the sentence which may be analyzed syntactically to determine the major determinant concepts. Today, still the phrase structured grammars are used instead of refined transformational grammar for its simplicity. On the way to capturing semantics, the first problem is ambiguities in meaning. When it is said, "*they are walking dogs*"- there are two semantics derivable depending on whether they refer to people or dogs. There have been many models for semantic analysis although none appears to be adequate. Kuno gave a list of unsolved problems in semantic theory.

1. The recognition of equivalence of meaning between "He bought the book from Amazon.com" and "Amazon.com sold the book to him".
2. The identification of correct transitivity of concepts. For example, "Dick is taller than Rich" and "Ed is shorter than Rich" means "Dick is taller than Ed". But "Rich met John" and "John met Kahn" does not mean that "Rich met Kahn".
3. The awareness of the common experiential world (e.g. a man becomes adult at 18 years of age, but a dog becomes adult at 2 years of age).
4. The recognition of degree of presence or absence of a given semantic feature. When one says. "toy is in a pen" means it is not a fountain pen, but a playpen.

Keeping these considerations in mind, it is skeptical if full semantics analysis will even be possible in future. At present, simpler structural analysis of phrases etc. are the methods being pursued in stead to further the effectiveness in indexing. The syntactic structure of key phrases does reflect the relations (being like semantic graphs) among components and thus serve as a compromise between semantics and syntax.

2.17 Criterion phrases:

A criterion phrase then consists of a set of English words, or concepts bound in a syntactic structure but for which the semantic interpretation is left unspecified. The syntactic structure is specified as a pre-stored frame, including concept numbers, syntactic indicators, and syntactic dependency relations that exist between the concepts.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

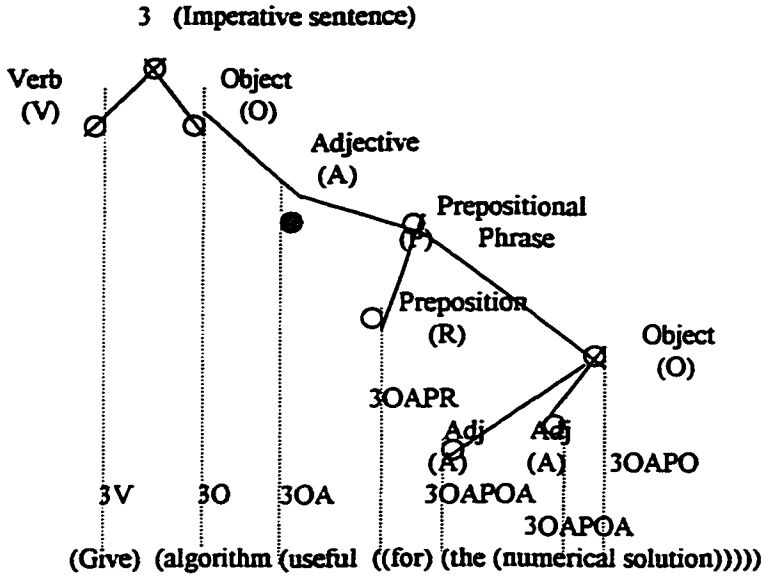


Figure 39: A modified dependency tree

Graphically, each criterion phrase can be shown as a syntactic dependency tree that shows the surface structure of the corresponding word string. Each node in the dependency tree is marked with a series of syntactic and semantic markers (syntactic indicators and concept numbers), and the syntactic dependency relations are indicated by the dependency tree structure. To search for the presence of syntactic phrases in a given document the following procedure must be followed:

1. The sentences extracted from the document must be parsed on the basis of a phrase-structured grammar to create a surface tree structure.
2. A criterion tree dictionary must be available for the given topic area which will contain all the important syntactic phrases.
3. A searcher process must try to match the surface tree structure in conjunction with the dictionary phrases.

In many systems, the analysis system, as mentioned in step 1 is equivalent to producing a syntactic dependency tree. Fig 39 shows an example of the structure of a syntactic dependency tree: The sentence parsed was, "Given the above problem, please give an algorithm which is useful for the numerical

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

solution". Each word has a syntactic code associated with it in which the number of characters specifies the level (e.g. node 3OAPOA has level 6 and node 3 has level 1), and the string specifies the connection and syntactic function. The string is of a form, αA where prefix α is the syntactic marker attached to the parent node of the current node and the single character A is the syntactic class of the current node (in other words, it is the syntactic function of the given word). Thus, the syntactic code 3OA indicates that the corresponding word (useful) is interpreted as an adjective, grammatically dependent on the node labeled 3O (algorithm) and is at the level 3. There are some dummy nodes in the syntactic tree to mark phrase and clause boundaries. A decomposition into phrase components is derivable from the syntactic trees, as before, by bracketing the words located in the same sub-tree.

In addition to the syntactic dependency structure for each analyzed sentence and to the syntactic markers attached to the text words, the concept numbers also can be added to the words. The dictionary contains symbolic structure which is similar to the dependency tree structure in that each phrase is identified by three types of information: the applicable concept number, the applicable syntactic indicators and the syntactic dependency structure. Four main classes of syntactic phrases are used: noun phrases, subject-verb relations, verb-object relations and subject-object relations. Approximately 15 different structure types are defined in the dictionary.

2.17.1 Criterion phrase format:

One syntactic phrase definition consists of six different types of specifications:

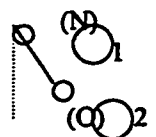
1) The name and serial number of the phrase.
2) The output concept number to be assigned to the phrase.
3) The nodes (or node) in the sentence to which these output concept numbers are to be attached.
4) The syntactic dependency relations between the nodes.
5) The syntactic indicators with each node of the phrase which must match the indicators of the document's sentence.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6) The semantic values (or the concept numbers) of each node in the tree that must match the concept numbers of each node from the document's sentence.

Ordinarily, in a syntactic phrase items 3,4 and 5 are combined together and the information is given with a given tree type number. The tree type number identifies the tree structure (i.e. how many nodes are there and how are these nodes arranged etc.). An example of a specification and the associated dependency tree types are shown in fig 41 below. The tree types 1, 3 and 5 result in the different trees as the diagram. This illustrates why the phrases are specified in terms of tree names, serial numbers, concept numbers attached to the nodes, and tree type numbers.

The tree name and serial number (a unique number for each syntactic dependency structure) are used to identify a syntactic phrase in the dictionary. Thus in the example shown here, phrase NUMERI is assigned serial numbers 87, 88 or 89 depending on if the phrase is of syntactic type 1, 3, or 5. The output concept number is the semantic marker that will be the identifier for the syntactic phrase. For example, when a phrase is detected in a document, this semantic marker is assigned to the document as an identifier along with a specific weight and thus will participate in defining the document concept vector. Also such an assignment of concept number makes it possible to embed a syntactic phrase within another one where in the outer phrase, the sub-tree for the inner phrase will be referred to by the concept number. An example will be "convergence of Bode-plot solutions". If there is already a phrase as "Bode-plot solutions", then the new phrase will be an associative tree of two nodes as in

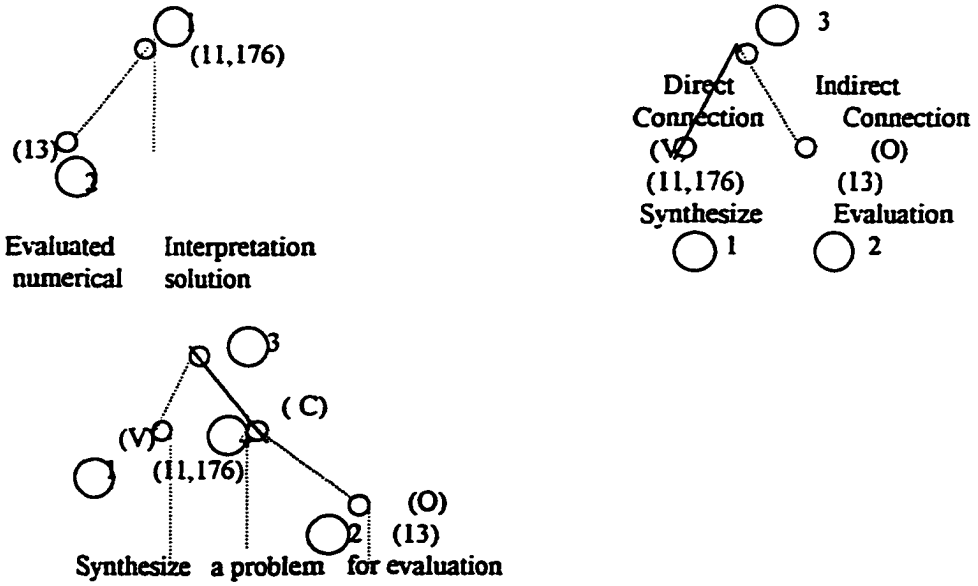


(Convergence) (Bode-plot solutions) this is concept no 71

Figure 40: Embedded phrases

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Specification: NUMERI (11,176)/(13)$1/87,3+,5+



Possible syntactic trees for concepts 13 and (11,176) of syntactic types 1, 3, 5

Figure 41:Sample syntactic phrase specification

The dependency connections between the various component nodes are also specified by referring to a syntactic tree type. But, also these connections can be defined by giving the parent node number for each node (except the root) and specifying the type of connection i.e. direct or indirect. A direct connection means that no intermediate node (or word) is allowed between the two connecting node. An indirect connection means any intermediate node (or word) will be disregarded. An example has been given in fig 42a. The last elements included in the specification are syntactic indicators and concept numbers.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Type 3	D3('V') / I3('O')		/	$3T1
Name of the defined structure.	Node 1 is directly connected to parent Node number 3 and has a 'V' syntactic marker.	Node 2 is indirectly connected to parent node 3 and has a 'O' syntactic Marker.	Node 3 is the parent node and has no marker.	Type 3 node with key node 1.

(a) Typical tree type specification

INFRET = 301.2.(26)/(53,114)$1/10,3+

INFRET = 301		.2.	(26) / (53, 114)		$1/10,3+
Name of the defined phrase	Output concept number	Key node is node 2 in stead of node 1.	Node 1 shows concept 26.	Node 2 matches concept 53 or 114.	Tree is being specified By type 1, serial 10, or Type 3, serial 11.

(b) A typical syntactic tree specification

INFRET=301d3('v')(26)/i3('o')(53,114)/$0/10,3T1

INFRET = 301		d3('v')(26) / i3('o')(53,114)		/	$0/10,3T1
Name of the phrase.	Output concept.	Dependency Connections and syntactic and Semantic indicators of node 1.	Connections and markers for node 2.	Vacuous Specifications for node 3.	No type given (as it is specified here), serial No 10; or tree type 3 with serial no 1.

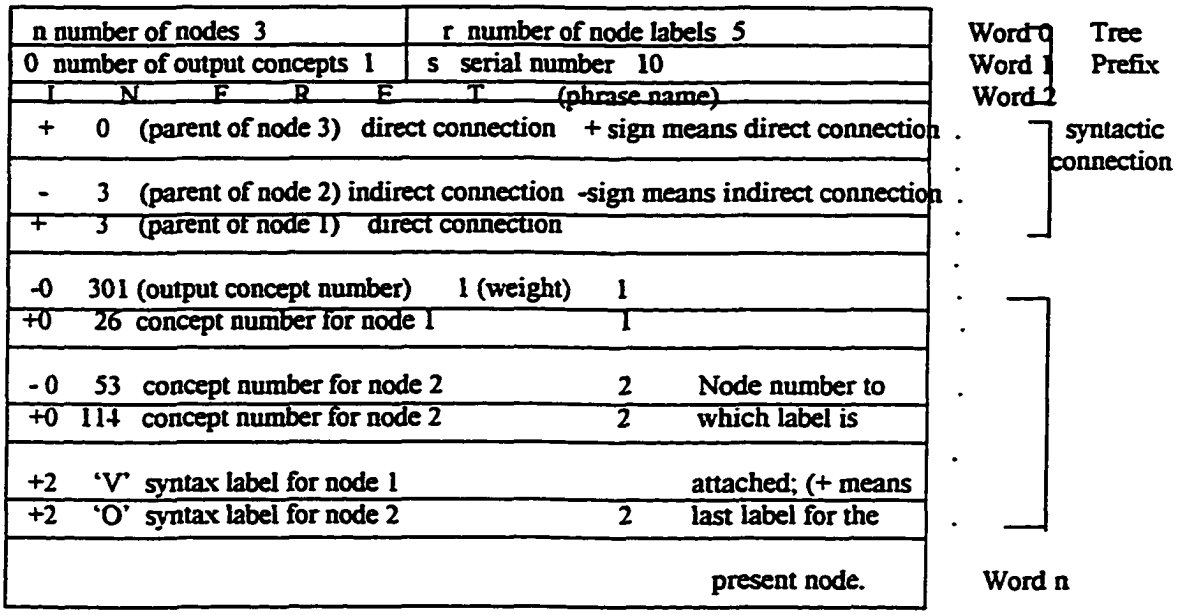
(c) typical phrase specification including syntactic type indications

Figure 42:Syntactic phrase and tree type formats

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In addition to the basic criterion phrase specification, it is also possible to specify syntactic tree types consisting of dependency relations, syntactic markers, and key nodes; furthermore, tree types and criterion phrases may be defined simultaneously as part of the same defining statement. Since each output concept number is normally assigned a weight, the weights can also be specified if different from a normal weight 1. Concept numbers can be shifted from node to node and the weights can be altered in the process.

The syntactic phrase formats can be very explicit as in fig 42. A typical tree type specification is shown in Fig 42a. Node 1 of the tree is specified to be directly connected (without any intermediary node) to parent node 3 and is marked with a verb ('v') syntax indicator; node 2 is indirectly connected to parent node 3 and is specified to have syntactic marker of 'object'. No parent is given for node 3 and has no syntactic marker. The type number and key node carrying the output concept number are given after the dollar symbol. Fig 42b presents a typical criterion phrase specification, where the dependency structure is denoted by tree types 1 and 3, corresponding to serial numbers 10 and 11 respectively.



Prefix -0 indicates 301 is a specific concept number (and not a node number whose concept is being attached to a new node)

Figure 43:Internal storage format for syntactic phrase INFRET

Type 3 D3('V')/I3('O')/$3T1

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Figure 43:Internal storage format for syntactic phrase INFRET

Type 3 D3('V')/I3('O')/$3T1

The output concept number to be attached to any matching phrase will be 301 and the sentence node to which this concept is to be attached is the correspondent of phrase node 2 (rather than corresponding node 1). Fig 42c combines specification of tree type 3 with a definition of the criterion phrase INFRET. For internal storage, a complete tree may be stored in a set of adjacent computer words as shown in fig 43.

Fig 43 shows the internal storage format to store the syntactic dependency tree for an example syntactic phrase "INFRET". The first three words contain total count of nodes in the tree, total number of node labels, number of output concepts, serial numbers and the phrase name. Next section shows all the dependency connections; next section shows output concept numbers; next section shows all the concept labels for all the components; next section shows all the syntactic labels.

2.17.2 Syntactic phrase processing

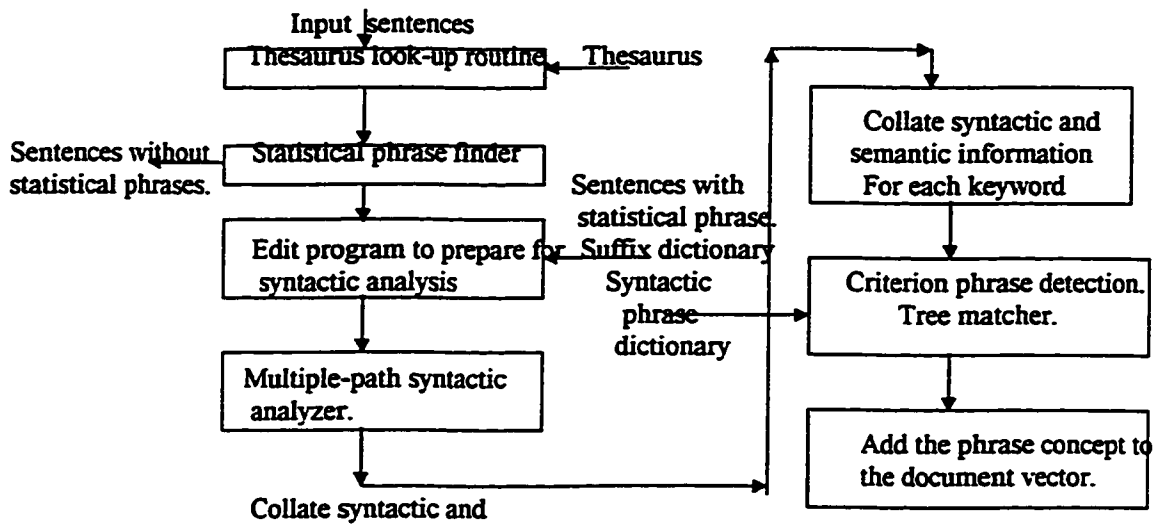


Figure 44:Flow-diagram for syntactic phrase processing

Fig 44 shows the flow-diagram for syntactic phrase processing. It may be seen that the criterion phrase process is used principally to eliminate from the document identifiers the statistical phrases that include components not properly related by specified syntactic dependency connections. A syntactic phrase

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

dictionary is used to store all the applicable phrases and automated procedures are available to update the phrase dictionary by deletion or addition of phrases.

A syntactic phrase is detected in a sentence if, for each node in the phrase there is a corresponding text node such that the following conditions hold:

1. No two phrase nodes correspond to the same node in the sentence.
2. The dependency connections between phrase nodes are reflected by similar dependency relations between the corresponding nodes in the sentence.
3. The correspondents of each phrase node satisfy the restrictions imposed by semantic and syntactic labels.

The matching process proceeds as follows. The first document sentence is matched with all trees in the syntactic phrase dictionary; then the second sentence is matched and so on. The matching phrase concepts are later added to the document concept vectors after their weights are ascertained.

The input sentence is fed to the syntactic analyzer which builds a list of keywords, their syntactic classes, syntactic role, the prediction pool etc. using the phrase structure grammar. From this list the modified dependency tree is built up starting with the leftmost path of the tree (the root comes at the top level and the other keywords hang from the root or its branches etc.). The character string representing a sentence structure code for a given word is examined one character at a time, starting with the leftmost character, and a path of the tree is built up. The path corresponding to a character string deviates from its predecessor at the first character that differs from the corresponding character in the preceding string: from that point on, a new path is followed in the tree structure. For example, if the last character string processed is 3OAPR (refer to fig 39), and the current string is 3OAPOA, the existing path is followed down to level 4 (upto 3OAP); then a new branch is created, so that the node coded 3OAPO becomes the twin of node 3OAPR and then node 3OAPOA becomes the leftmost child of node 3OAPO.
The actual tree generation methodology is presented in fig 45 for a simple sentence.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Sentence structure code	Present string	Last node vector	Parent vector	Resulting tree structure
INFORMATION 1SA	-	-	-	-
	1	1	0	(1) 1
1SA	1,S	1,2	0,1	(1) 1 (2) 2
1SA	1,S,A	1,2,3	0,1,2	(1) (S) (A)
RETRIEVAL 1S	1, S, A	1, 2, 3	0,1,2	(1) 1 (S) 2 (A) 3
1S	1,S,A	1,2, 3	0,1,2	(1) 1 (S) 2 (A) 3
IS 1V	1, S, A	1, 2, 3	0, 1, 2	(1) 1 (S) 2 (A) 3
1V	1, V, A	1, 4, 3	0,1,2,1	(1) (S) (A)3 4(V)
USEFUL 1C	1, V, A	1, 4, 3	0, 1, 2, 1	1 (1) (S) (A)3 4(V)
1C	1, C, A	1,5, 3	0, 1, 2, 1, 1	(1) 1 (S) 2 (A) 3 4 (V) 5(C)

Figure 45:Creation of sentence tree from syntactic structure code

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Three vectors are used in the process, consisting of the character string presently being scanned and of the last node and parent vectors. The ith element of the last node vector stores the number of the node last created at level i; this vector is updated whenever a new node is created, and its length equals the number of levels of the tree structure. The ith element of the parent vector stores the node number of the parent of node i. The changes in the three vectors are shown in details in the above fig 45 and the resulting sentence tree structures are shown in the rightmost column.

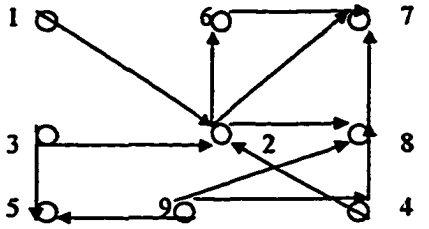
2.17.3 Syntactic tree matching

The syntactic tree matching becomes important for matching the document sentence syntactic tree in conjunction with the trees in the phrase dictionary. There are several basic types of structure matching methods as *node-by-node*, *piece-by-piece* and the *topological* method. In the *node-by-node* method, the nodes of the two trees (or structures) are matched one at a time, until a complete match is done or a disparity occurs when one has to backtrack upto a point of agreement and try another combination. In piece-by-piece method, a complete structure is broken into a set of dictionary basic substructures which are then matched as a whole. But till today, there is no standard method of decomposing structures into substructures. Node-by-node method works well for small graphs, but becomes too complex for large graphs (with many nodes). The topological matching method is good for graphs of arbitrary complexity.

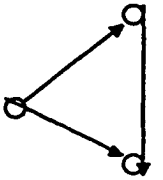
2.17.3.1 General topological structure matching

This matching procedure is quite general in the sense that it is independent of specific ordering of the nodes, presence or absence of certain substructures. No backtracking is needed and this method can also be used to match partial matches. The method can be well illustrated using a large graph called *document structure* (which plays the role of document sentence tree) and the smaller query structure represents a criterion phrase (in fig 46). The large graph has nodes numbered arbitrarily from 1 to 9 and the smaller graph has nodes a to c and the connection pattern of the nodes is shown as the binary connection matrices in fig 47.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

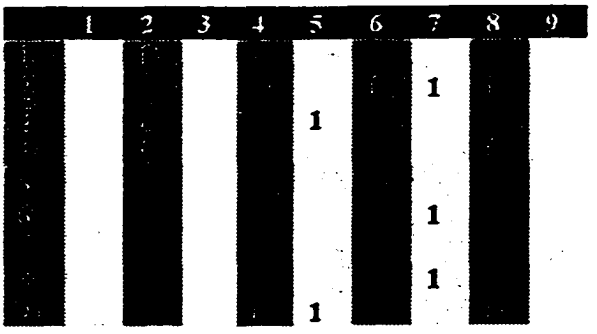


(a) Document structure

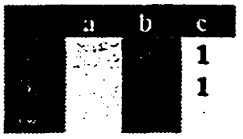


(b) Query structure

Figure 46:directed graphs



(a) Document structure



(b) Query structure

Figure 47:Connection matrix for graphs in fig 46

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 2.3: Correspondences formed for the graphs of fig 46.

Set number	Criterion for set formation	Corresponding sets
I	Outward degree 1	{b} ⊆ {1, 2, 3, 4, 5, 6, 8, 9}
II	Outward degree 2	{a} ⊆ { 2, 3, 4, 9}
III	Inward degree 1	{b} ⊆ {2, 4, 5, 6, 7, 8}
IV	Inward degree 2	{c} ⊆ {2, 5, 7, 8}
V	Partition	{a} ⊆ {2, 3, 4, 9}
VI		{b} ⊆ {2, 4, 6, 8}
VII		{c} ⊆ {2, 5, 7, 8}
VIII	Outward connection, sets V	{b, c} ⊆ {2, 4, 5, 6, 7, 8}
IX	sets VI	{c} ⊆ {2, 4, 7, 8}
X	Inward connection, sets VI	{a} ⊆ {1, 2, 3, 4, 9}
XI	sets VII	{a, b} ⊆ {1, 2, 3, 4, 6, 8, 9}
XII	Partition	{a} ⊆ {2, 3, 4, 9}
XIII		{b} ⊆ {2,4, 6, 8}
XIV		{c} ⊆ {2, 7, 8}
XV	Inward connection, sets XIV	{a, b} ⊆ {1,2, 3, 4, 6, 8, 9}
XVI	Partition	{a} ⊆ {2, 3, 4, 9}
XVII		{b} ⊆ {2, 4, 6, 8}
XVIII		{c} ⊆ {2, 7, 8}
XIX	Assignment	{a} = {2}
XX	Outward connection, sets XIX	{b, c} ⊆ {6, 7, 8}
XXI	Partition	{a} = {2}
XXII		{b} = {6, 8}
XXIII		{c} = {7, 8}

The connection matrices are equivalent to the directed graphs from information point of view. The objective is to use the properties of the nodes and/or branches of the graphs and generate pairs of sets which must match in order for the graphs to match. The following properties are useful for this purpose:

1. The k-th order outward (or inward) degree of the nodes, that is the number of nodes reachable from a given node by outgoing (incoming) paths of length k.
2. Labels or identifiers that may be associated with nodes or branches.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3. The connectivity patterns of sets of nodes, that is, the nodes reachable from a given set of nodes by paths of length k.

The initial set correspondences are given in lines I to IV in the table. Set II, for example, is constructed by noting that the set of all nodes of outward degree 2 in the query structure must correspond to the set of all nodes having at least outward degree 2 in the document structure, because a node with two outgoing branches in the query structure can not match a document node with one outgoing edge; but the two branches can be matched with any two branches of a node with outward degree greater than 2. The only node of outward degree 2 in fig 46b is a and the nodes with outward degree >= 2 in 46a are 2, 3, 4, 9.

At this point, it is necessary to generate smaller sets from the ones shown on lines I to IV of table 2.3. This is done by forming sets using joint properties of sets. For example, the set of query nodes that have both outward degree 1 (criterion for set I) and inward degree 1 (criterion for set III) must correspond to the set of document nodes with outward and inward degrees at least equal to 1. The only query node with the required property is b (both inward degree and outward degree 1); four nodes possess the given properties, including 2, 4, 6, 8 in the document graph. This correspondence shows as set VI in the table 2.3. Thus the size of the correspondents is reduced from 7 to 4 for node b (in set VI). Thus the partitioning process generates three new sets (V, VI, VII) all obtained by intersection operation. New sets (VIII, IX, X, XI) can be added by outward and inward connections of length 1 from the sets V, VI and VII. For example, an inward connection to node b produces node a in the query graph; the inward connection to nodes {2, 4, 6, 8} of degree 1 or more produces set of nodes {1, 2, 3, 4, 9}. Now the partitioning process is done on sets V through XI. For example, set VII says, $\{c\} \subseteq \{2, 5, 7, 8\}$. Again set IX says, $\{c\} \subseteq \{2, 6, 7, 8\}$. Now an intersection of these two gives $\{c\} \subseteq \{2, 7, 8\}$.This way sets XII through XIV are obtained. As the set for {c} has changed, again the inward connectivity test is performed. That produces set XV. Another round of partitioning produces sets XVI, XVII, XVIII. As these partitions are exactly the same as before, so there is no need to do any connectivity test. At this point, one can make an assignment which needs to be verified later. Such an assignment has been done in set XIX

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

as {a}={2}. The outward connection of length 1 on set XIX gives {b, c} $\subseteq$ {6, 7, 8} (set XX). Intersection between sets XVI through XVIII and XIX through XX produces the sets XXI, XXII, XXIII. Now one possible assignment is

a = 2, b = 6 and c = 7 or a=2,b=8 and c=7 and both are valid assignments.

Instead for set XIX, another assignment could be made as {a} = {3}. Then outward connection on length 1 gives {b, c} = {2, 5}. Partitioning this with sets XVI, XVII, XVIII will produce {b}={2}, but {c} will be empty set. This violates the starting principle - there has to be at least one corresponding node. Thus this incompatible assignment is discarded.

However, there are other valid assignments as:

{a} = {4} and {a} = {9}

{b} = {2} {b} = {4}

{c} = {8} {c} = {8} etc. All these four mappings can be verified by looking at the graph in fig 46a.

A flow-chart for the graph matching procedure is given below in fig 48. This procedure is iterative because once the sets are generated, then a partitioning takes place followed by generation of new sets etc. Such iteration continues in the procedure unless one of the three situations happen:

1. The pair-wise mapping is reduced to one-to-one mapping thus resulting in at least one total match.
2. An incompatibility arises between pairs of sets due to a cardinality violation.
3. Repeating this steps does not result in any new sets but creates the old sets. At that time, an arbitrary assignment needs to be made.

A cardinality violation is detected if, for example a set A is found in a set B which has fewer members, then the two structures can not match; it is an error. In such situation, one can backtrack if there are alternate paths or the result is a mismatch.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

But the above algorithm needs to be modified for comparing syntactic dependency trees firstly because between any two nodes in the query structure there may be any number of dummy intermediate nodes. For example, when dummy intermediate nodes are introduced in the query structure of fig 46b, the new structures as in fig 49b and fig 49d are also created where the broken lines indicate that indirect connections are also possible. Now a topological comparison of fig 49b with a partial sub-graph of fig 46a is shown in fig 49a or another comparison between fig 49d and fig 49c prove that there are other

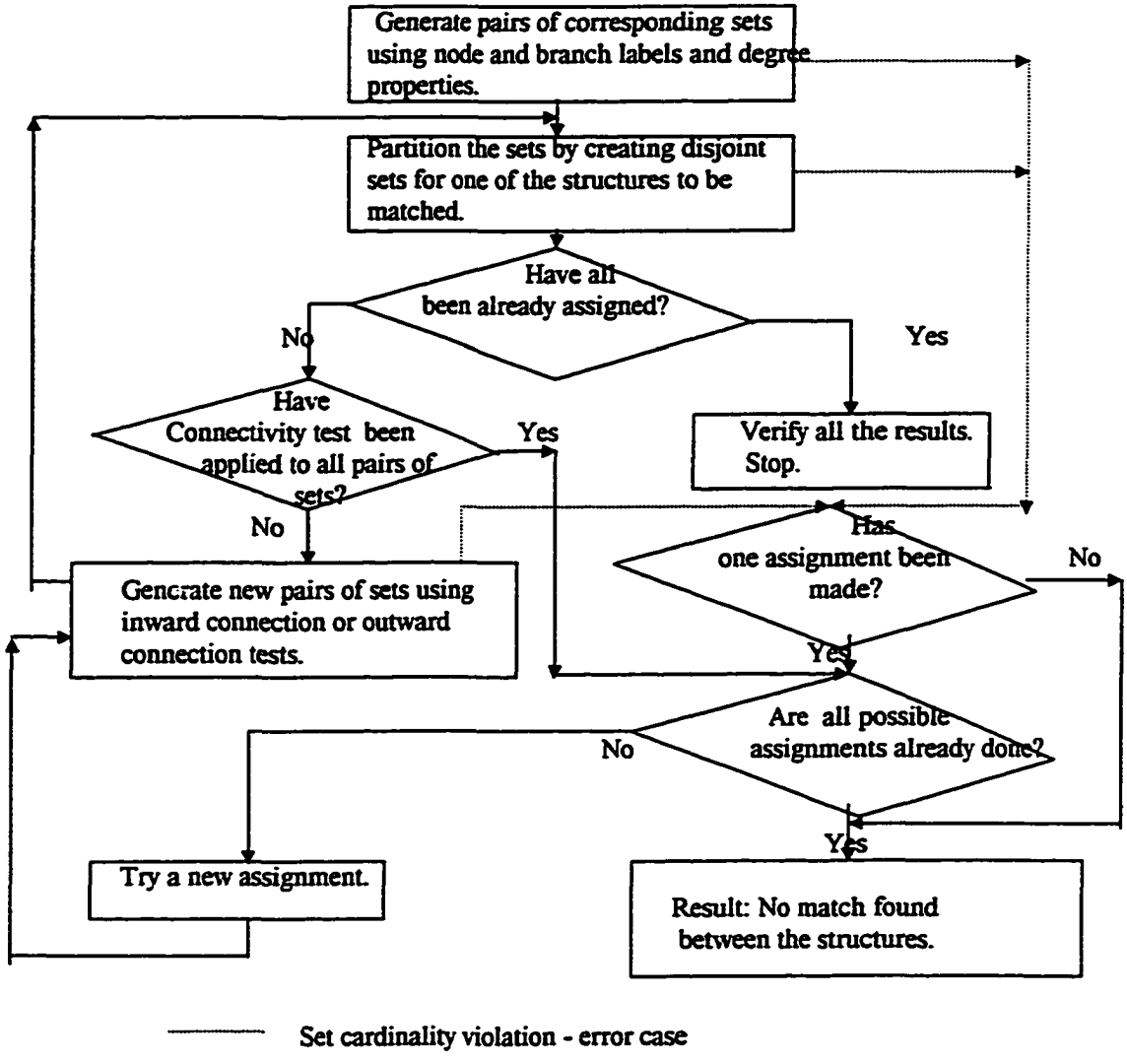
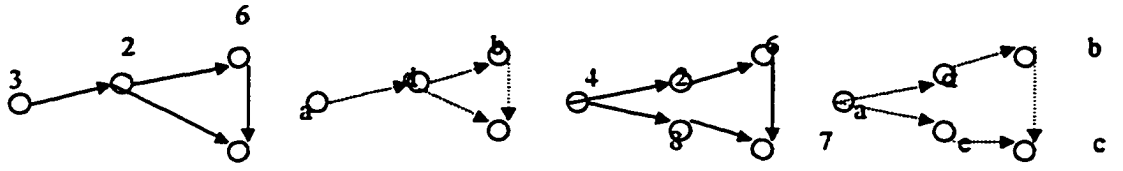


Figure 48: Flow-diagram for graph-matching procedure

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



(a) Partial sub-graph of document structure of fig 46a.

(b) Modified query structure from fig 46b.

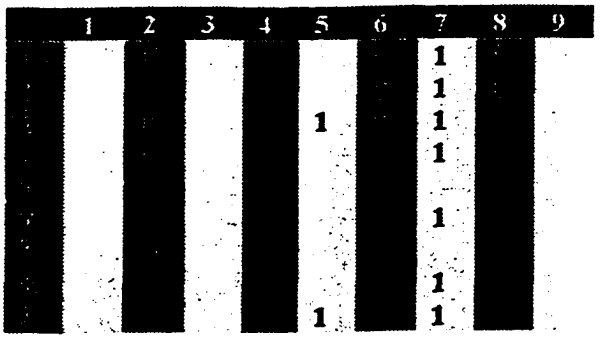
(c) Partial sub-graph of document structure from fig 46a.

(d) Modified query structure from fig 46b.

Figure 49:Topological matching of partial substructure

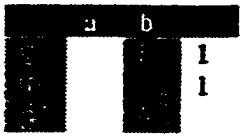
substructure matches that is possible between the query structure and document text apart from the previous 4 matches. Even further partial substructure matches (which are not shown in figure) are also possible.

To determine if two graphs can be matched when dummy intermediate nodes are present, it is no more always necessary to know if two nodes are directly connected or not, but it is necessary to know if a route (indirect or direct) exists between two nodes or not. Thus the connection matrices of fig 47 must be changed to path matrices in fig 50, in which the ij-th element is 1 whenever a path exists from node i to node j. Such path matrices can be derived by matrix multiplication as a sum of powers of the corresponding connection matrices. If the connection matrix is C, then C gives the paths of length 1, C.C gives all the paths of length 2, C^3 gives all the paths of length 3. But no of lengths does not have to exceed the no of edges in the graph. So, it is given by $C+ C.C + C.C.C + ...+ C^n$ where n is the no of edges in the graph. This path matrix is given in fig 50 below.



(a) Path matrices for Dictionary structure

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

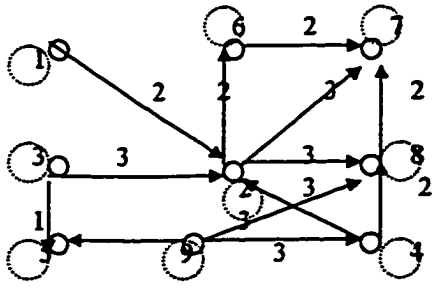


(b) path matrices for query structure

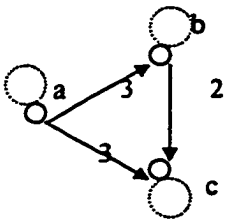
Figure 50:Path matrices (computed from the connection matrices)

Using the previously described algorithm with this new path matrices will generate the isomorphisms as in table 2.3 and a few new substructures as in fig 49.

Another change in the matching procedure comes up because so far in the assumption the branches of the trees were unlabeled. But in the document analysis, the relations come up of certain types for example: *associative*, *predicative* and *consecutive* relations etc. The directed graphs must be redrawn with the relation codes included as the branch labels as in fig 51. Now the matching procedure must include a topological match in the relation codes. The binary connection matrix may be replaced by a numeric branch label matrix as shown in fig 52 where the 1s are replaced by a numeric value n meaning the different relation codes such as relation type 1, relation type 2, relation type 3 etc. In the set generation, partitioning and connection methodology procedure also, all operations (e.g. outgoing branch relation type 1 etc.) must be done with respect to the branch labels, and now for assignments, not only must the nodes match but also the branch labels must match. For example, the initial sets for the modified matching procedure can be as given in table 2-4 below. The branch labels make it possible to have a large number of sets and the set partitioning procedure results in the following correspondences.



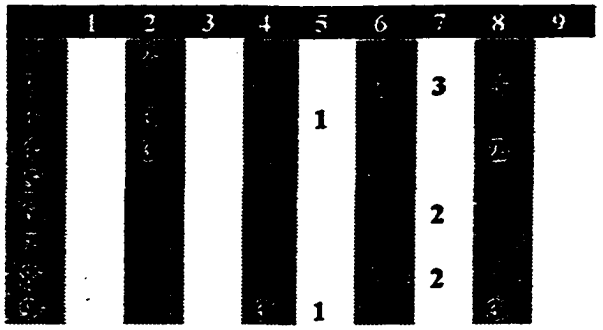
(a) Document structure



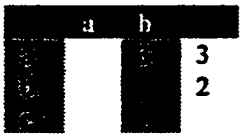
(b) Query Structure

Figure 51:Directed graphs including branch labels

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



(a) Document structure



(b) Query structure

Figure 52:Branch label matrices

Table 2-4: Set correspondence for graphs

Set number	Criterion for set generation	Corresponding sets
I	Outward degree 2 (or greater) and outward branch label 3,3.	{a} ⊆ {2, 9}
II	Outward degree 1 (or greater) and branch label 2.	{b} ⊆ {1, 2, 4, 6, 8}
III	Inward degree 2 and branch label 2,3.	{c} ⊆ {2, 7, 8}
IV	Connections from set I with branch label 3.	{b, c} ⊆ {4, 7, 8}
V	Connections from set II and branch label 2.	{c} ⊆ {2, 6, 8, 7}
VI	Connections into set III and branch label 2.	{b} ⊆ {1, 6, 8, 4}
VII	Connections into set III and branch label 3.	{a} ⊆ {3, 4, 2, 9}

{a} ⊆ {2,9}
{b} ⊆ {4, 8}
{c} ⊆ {7, 8}

Now assignment of {a} to either node {2} or node {9} produces the following two one-to-one mappings as follows:

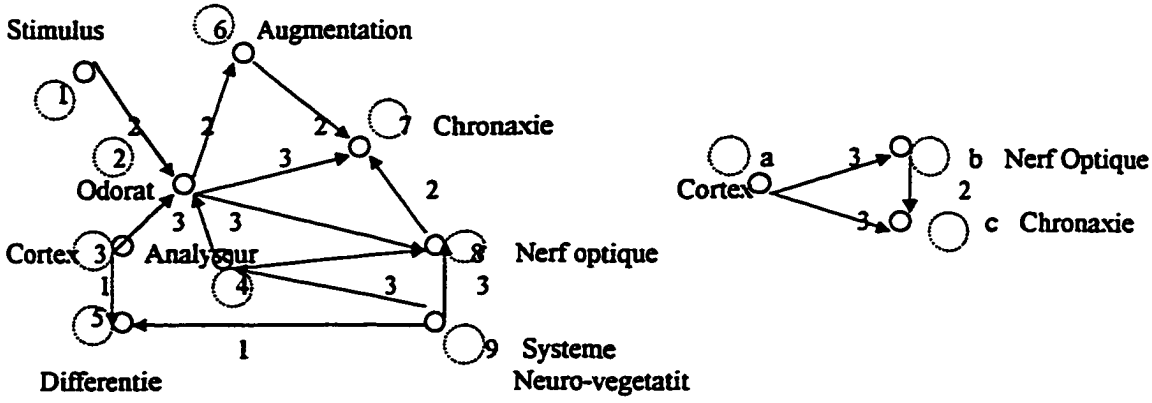
{a} ↔ {9} and {a} ↔ {2}
{b} ↔ {4} {b} ↔ {8}
{c} ↔ {8} {c} ↔ {7}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

This may be verified from fig 51 that the above two mappings are the only ones that follow the label restrictions.

As another extension, the nodes in the actual document graph will carry some syntactic & some semantic labels. In order to match it is necessary that the node labels also correspond (if not exactly match).

For example, a node label matrix may be represented in the following table 2-5 for the graph that is shown in fig 53.



(a) a typical Syntol document graph (b) typical syntol query graph

Figure 53:Syntol document graph and Syntol query graph

Table 2-5 Set correspondence for graphs in fig 53 using direct connections

Set number	Criterion for set generation	Corresponding sets
I	Node label *cortex*	{a} ⊆ {3}
II	Node label *nerf optique*	{b} ⊆ {8}
III	Node label *chronaxie*	{c} ⊆ {7}
IV	Direct connections from set I with branch label 3	{b, c}⊆{2}

Since all the nodes are labeled, an immediate correspondence is established between the nodes of the two graphs under consideration (sets I, II and III of table 2-5). One has to determine if the connections and branch labels are preserved. Sets IV of table 2-5 indicate an error, as node a has two outgoing branches

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

with branch label 3 and the corresponding node 3 has one outgoing edge. Since a set containing two elements can not exist in a set of one element, this is an erroneous correspondence. So the test fails.

So it is required to relax the restrictions by considering indirect connections and indirect nodes. The path matrix of fig 52 is now used to verify that complete paths and path labels are preserved by the correspondence I the node labels. The set correspondences, as shown in table 2-6 demonstrate that to each outgoing and incoming labeled path in the query structure there corresponds a path with similar properties in the document structure. When the partitioning process is carried through, the query graph is seen to match the document graph with the following correspondences between the paths:

Path (a) to (b) corresponds to path (3) to (2) to (8)

Path (b) to (c) corresponds to path (8) to (7)

Path (a) to (c) corresponds to path (3) to (2) to (7)

Table 2-6 Set correspondences for graphs of Fig 53 using indirect connections

Set number	Criterion for set generation	Corresponding sets
I	Node label *cortex*	$\{a\} \subseteq \{3\}$
II	Node label *nerf optique*	$\{b\} \subseteq \{8\}$
III	Node label *chronaxie*	$\{c\} \subseteq \{7\}$
IV	Paths from I with branch label (3)	$\{b, c\} \subseteq \{2, 7, 8\}$
V	Paths from II with branch label (2)	$\{c\} \subseteq \{7\}$
VI	Paths into III with branch label 2	$\{b\} \subseteq \{1, 2, 4, 6, 8\}$
VII	Paths into III with branch label (3)	$\{a\} \subseteq \{2, 3, 4, 9\}$
VIII	Paths into II with branch label (3)	$\{a\} \subseteq \{2, 3, 4, 9\}$

An adjustable procedure for the comparison of query and document structures can now be outlined. The process uses the same matching algorithm throughout and is modified only by altering the matrices that represent the connection patterns and the branch or node labels. Clearly the exact strategy to use in altering the connection matrix may depend on the type of document collection being processed. When the graph matching is applied to syntactic dependency trees, the following search strategy may be used:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1. Given a specified node of the criterion phrase, all the sentence nodes that have matching syntactic and semantic labels are selected.
2. From among the sentence nodes that obey the restrictions of part 1, some subset may be selected whose direct (or indirect) connection pattern is identical with the connection pattern of the corresponding nodes in the criterion phrase.

For example, there is a syntactic phrase in fig 54 and there is a the syntactic tree for document sentence of fig 55. Here, both syntactic and semantic labels for nodes (7) and (b) and of nodes (9) and (a) will match fine. But there is a path from node (a) to (b) in the syntactic phrase, whereas such path does not exist from node (9) to node (7) in fig 55.. Hence the matching fails in this figure. But it can be verified that the syntactic phrase in fig 54 will easily match the tree in fig 56.

Finally, methods can be used to adjust the matching process. First, the matching algorithm is flexible, because node and branch labels may or may not be taken into account, and direct as well as indirect connections can be used. Also the syntactic phrase dictionary should be updated to include larger variety of sentence structures.

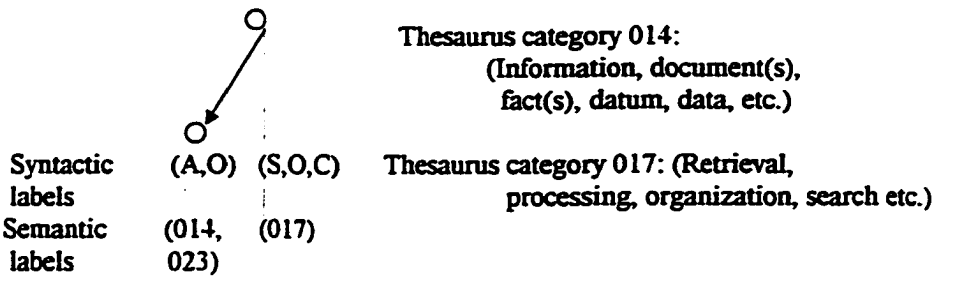
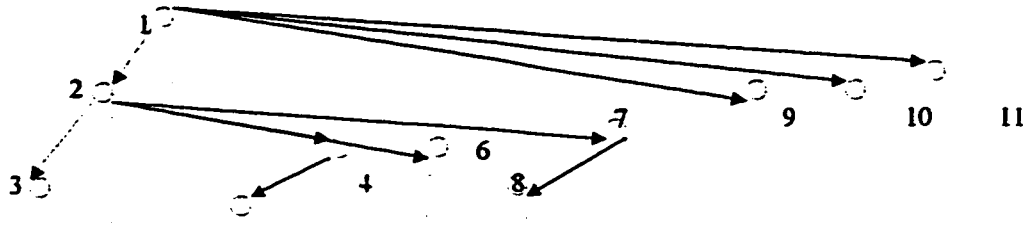


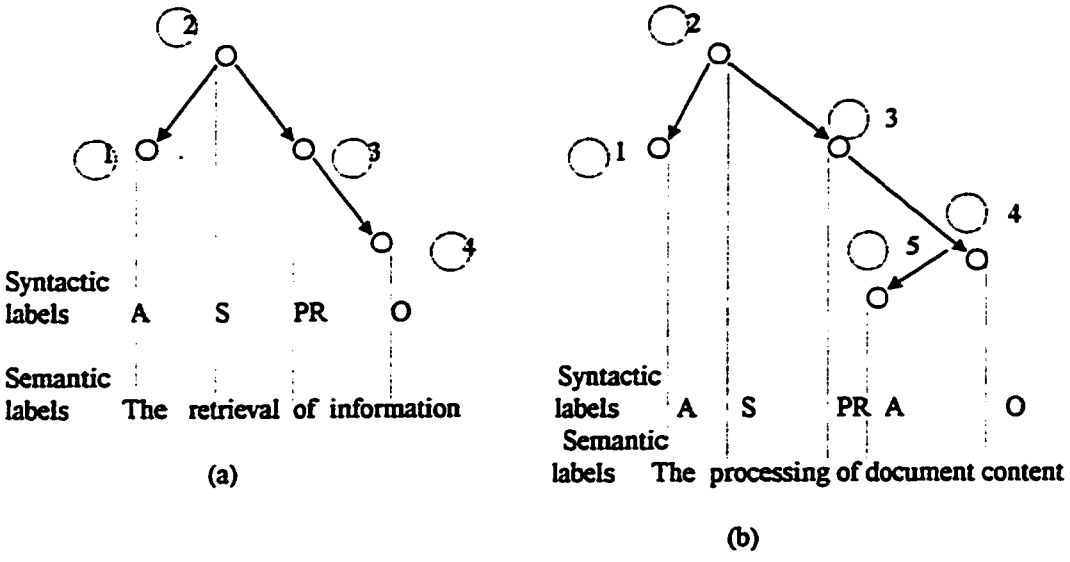
Figure 54:Typical syntactic phrase

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

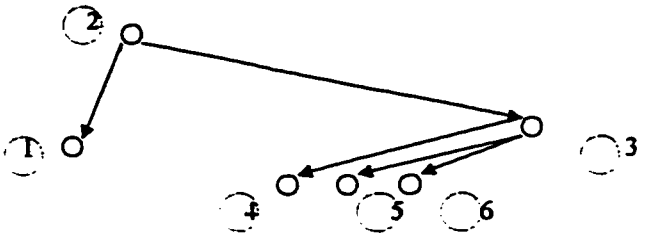


Syntactic labels											
	R	8	1	A	S	V	A	O	S	V	C
Semantic labels											
Because	-	-	the	text	contains	secret	information	retrieval	is	vital	

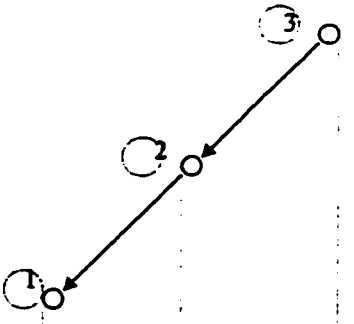
Figure 55:Typical document sentence syntactic tree



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Syntactic labels S V A + A O
Semantic labels He discusses information and document retrieval
(c)



Syntactic labels A A S
Semantic label Document file organization
(d)

Figure 56:{(a),(b),(c),(d)} Tree structures matching the syntactic phrase of fig 54.

2.17.3.2 Node-by-node tree matching

Since the criterion phrase contains only a few words (or a few nodes), the topological graph matching process, which is a structure-oriented method may not be the most efficient method. Instead a value-oriented method that counts on classification of nodes based on their values may be worth-trying. The structural correspondence may be verified in the second step.

In the first step, given a criterion tree and a structural dependency tree for the document text one has to generate a list mapping each node of the criterion tree into a list of nodes in the sentence tree with

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

corresponding syntactic and semantic labels. To start, the node-label assignments of the criterion tree are built and then a node correspondence matrix is built. An example node-label assignment matrix is given in table 2-7 for criterion phrase as in fig 57 below.

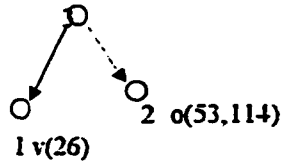


Figure 57:Example criterion phrase tree for phrase INFRET

Table 2-7 Sample node-label assignment

Label	1	2	3	4	5	6	7	8	9
'v'	1	0	0	1	0	0	1	0	1
'o'	0	1	1	0	0	0	0	1	0
26	1	0	1	0	1	0	1	0	0
53	1	0	1	0	0	1	0	1	0
114	0	1	0	1	1	0	0	0	1

In this example, the sentence tree is a hypothetical one. In the table, it shows that nodes 1, 4, 7 and 9 of the sentence tree carry the label 'v'. Also node 2, 3 and 8 correspond to label 'o' of the syntactic tree. For the concept numbers, 26 is found in nodes 1, 3, 5 and 7 in the sentence tree. This node-label assignment matrix is converted into a node correspondence matrix where each row is a dependent node in the criterion tree carrying the syntactic and semantic labels. The sentence node numbers are found by doing a logical multiply between two different elements on the different rows for syntactic label and semantic label. For example, take node ('o', 114) and sentence node 3; the two entries in rows 'o' and 114 are 1 and 0; so the product gives 0; thus the correspondence matrix is built.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

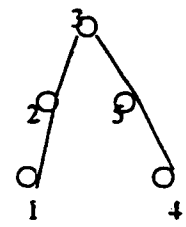
Table 2-8 Node correspondence matrix

Criterion phrase node	sentence node numbers 1	2	3	4	5	6	7	8	9
1('v',26)	1	0	0	0	0	0	1	0	0
2('o',53)	0	0	1	0	0	0	0	1	0
2('o',114)	0	1	0	0	0	0	0	0	0

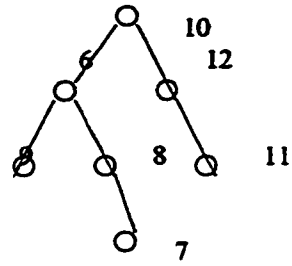
The node correspondence matrix is now used along with the *parent vectors* for the two tree structures to check if the connection patterns also match or not. The first step in this structure matching process is in choosing the criterion phrase node with the fewest possible correspondents and assigning to it one possible correspondent in the sentence tree. As the leaf nodes have more labels, so they will have less sentence node correspondents. So an assignment may be chosen out of these and carried on. Then an attempt is made to assign all the parent nodes (up and up) until the root. First a check is done if the parent of the correspondent node is a correspondent of the parent of the node under consideration. If it is and the parent node has not been already assigned then one of the remaining correspondents is assigned to it. This way, the chain is followed upward until the root. At this point, the next unassigned criterion phrase node (with the fewest correspondents set) is assigned a correspondent node and another path is traced upwards. This goes on till there is any unassigned node in the criterion tree. If at any time, it is not possible to assign all ancestors after a given node n was temporarily assigned or the parent node is already assigned to a correspondent that is not the same as the parent of the correspondent, then the temporary assignment is canceled and backtracking is done.

The algorithm becomes slightly complicated due to the inclusion of indirect connections. In that case, all parents on any higher level must be checked at each point and it will be okay if the given criterion tree node is the grandparent or any other ancestor of a previously assigned node. A sample set of node correspondences is given in fig 58c for the syntactic phrase and sentence nodes of fig 58a and 58b respectively. The progress of the algorithm is given in table 2-9 which shows the 2 backtracking steps ending in the final assignments.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



(a) sample criterion phrase structure



(b) sample sentence tree structure

	Criterion nodes	Corresponding sentence nodes
Nodes ordered by increasing number of correspondents	1	{9, 7, 8}
	4	{7, 12, 11}
	2	{6, 8, 7, 10}
	5	{12, 9, 10, 7, 8}
	3	{10, 11, 12, 7}

(c) Node correspondence

Figure 58:Example for node-by-node tree matching

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 2-9 node-by-node tree tracing

Step 1	Unassigned node with fewest correspondents is 1	
	Assignment	1→ 9
	Check: is 2← 6 okay √	
	Assignment	2←6
	Check: is 3←10 okay √	
	Assignment	3←10
Step 2	Unassigned node with next fewest correspondents is 4	
	Assignment	4→7
	Check: is 5←8 okay √	
	Assignment	5←8
	Check: 6 (parent of 8) was already assigned; is it assigned to 3 (parent of the parent)? NO ×	Cancel
Step 3	Attempting to assign another correspondent for node 4.	
	Assignment	4→12
	Check: 10 (parent of 12) already assigned; is 10 correspondent of node 5? NO ×	Cancel
Step 4	Attempting another correspondent for node 4.	
	Assignment	4→11
	Check: is 5← 12 okay √	
	Assignment	5←12
	Check: 10 already assigned; is its image the parent of 5 okay √	
	Assignment; no check	3←10
Step 5	All assignments done; Results completed. To generate another set of assignments of correspondences, start wit another assignment.	1→7
	.	
	.	
	.	
	.	

2.17.4 *Quality*

An automatic criterion phrase is expected to perform well when compared with a manual analysis system. The flexibility of it renders extensive matching which results in higher recall. The flexibility is added by the four following features:

1. Word-stems rather than words are used for thesaurus look-up.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2. Concept numbers are attached to the phrase nodes in stead of word or word-stems.
3. Numerous syntactic relations are supported between the nodes of the criterion tree thus allowing many

syntactic forms.

4. When the dependency connection between two concept numbers is an independent connection, any extraneous words or function words will be accommodated between the designated concepts, thus increasing the flexibility in matching.

Thus it is possible to match semantically equivalent phrases as long as the same order of the important sentential parts (like subject, verb, object etc.) are retained. Differences due to addition or deletion of word will still match; shift from noun to verb constructions, use of synonyms, multiple subjects or verbs do not interfere with the match. But at the same time some structures will not be matched by the phrase matching procedure. Constructions such as, active-passive transformations, synonymous phrases, transitive semantics etc. will not be accepted by the phrase matching procedure. Sometimes the rigidity of the syntactic classes do not permit many statistical phrases which rightly identify the document. The experiments indicate that the precision decreases with syntactic phrase matching compared to statistical phrase matching.

2.18 Algebraic abstract model for Retrieval

In simple terms, the basic models of information retrieval is fairly simple. Each document is identified by a vector of document identifiers and so is with the query. These identifiers are also called concepts or keywords. While searching , the document identifiers are compared with the query identifiers. In addition some auxiliary restrictions also may be added. Abstractly speaking, the query retrieval can also be seen as a transformation from a set of keywords to a set of document descriptors or as a mapping from the space of all keywords sets to the space of all retrievable document sets. Some set-theoretical models (using sets and lattice theory) of retrieval are analyzed here with emphasis on those models where both the keyword space and the document space are partially ordered sets. Also the problem with the use

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of negation in the formulation will be analyzed with an extension of the set-theoretical model to those situations where relations are recognized between individual concepts where the model based upon sets and transformations are replaced by abstract graph and tree manipulations.

2.18.1 Inclusive information retrieval

Basically speaking, the fundamental data of information retrieval theory is provided by the relations that exist between the sets of document descriptions and the set of documents. There are a few important things to remember while considering the particular model.

1) The collection of items may be static in the sense that a well-defined complement lies for each set of items or it may change with time thus affecting the set operations as complements.
2) The query space may be identical with the space of document descriptions in such a way that queries are built using terms which are used to identify the documents or otherwise the query formulation may not be subject to the same restrictions as applied by document descriptions.
3) Relations may be defined between document and request identifiers, or on the opposite, the identifiers may be unstructured and independent of any other identifier.

These are the questions to be answered to decide on a model based upon the requirements of the site.

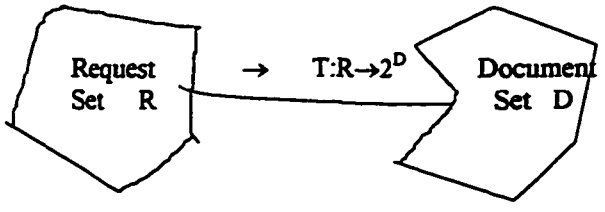


Figure 59:Request-Document mapping

Assume that in the system, a set R of information requests will be used to retrieve subsets of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of documents from a set D. The set R may be called the query set and document set D may be assumed to static. So formally the retrieval system is a triplet of document D and query language R and a mapping $T: R \rightarrow 2^D$ where T is called the retrieval function. The set of documents T(r) is said to be retrieved for a query $r \in R$ so that $T(r) \subseteq D$. Thus T maps a query r into a subset of D as shown in fig 59.

Now this information retrieval system is partially ordered if R is a partially ordered set i.e. a transitive and anti-symmetric relation $\geq$ is defined on R. If each query r is formed as a subset of terms from a static set of identifiers, then a partial order is automatically defined among the possible set of identifiers For example if there are 3 elementary terms (a_1, a_2, a_3) that may be combined (or dropped) to form a request, then the possible requests will be ordered in a lattice as in fig 60.

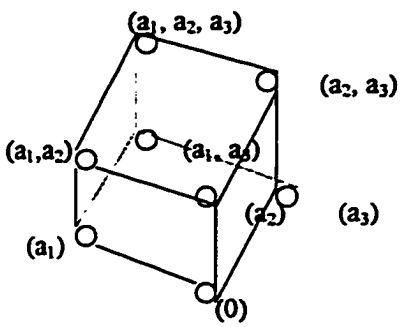


Figure 60:Lattice elements as subsets of the set (a1, a2, a3)

An inclusive information retrieval system can be defined to be a partially ordered information retrieval system, such that for all $r, s \in R$ $r \geq s$ means $T(r) \subseteq T(s)$ where symbol $\subseteq$ means set inclusion operator.

Normally, a standard keyword system is inclusive, because more keywords are supplied in the query or in other words the longer subset r is, the fewer the retrieved document set T(r) is. Also thus, if there are two requests r and s such that r contains all the elements of query s and may be some more then all the documents retrieved for r (T(r)) will be included in the document-set retrieved for query s although the document-set retrieved for query s may have some extra documents also. In other words T(s) may be larger or equal to T(r).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Let S be a subset of a partially ordered set R, and let r, s $\in$ S; then it may be possible to define the lub {r, s} (called join, denoted by r$\vee$s) and the glb {r, s} (called meet, denoted by r$\wedge$s). If S is a lattice defined by (S, $\wedge$, $\vee$) which consist of elements representing the subsets of a set, such that the partial order defined on the lattice is set inclusion, the join and meet operations become equivalent to union and intersect operations. These join and meet operations can be thought of as binary operators defined for two subsets r, s

for which lub{r, s} and glb{r, s} exist. Since the retrieval function T maps requests into elements of 2^D and the subsets of D form a lattice under set inclusion, for any pair of document sets X, Y $\in 2^D$, one can say

$X \wedge Y = glb\ (X, Y) = X \cap Y$ and

$X \vee Y = lub\ (X, Y) = X \cup Y$

Now when the query space R is a partially ordered set, it is interesting to note the relation between the order defined on the query space and the corresponding order in the document space D.

Theorem 1. In an <u>inclusive</u> information retrieval system, the image under T of a decreasing chain in R is an increasing chain in 2^D, and the image of any increasing chain in R is a decreasing chain in 2^D.

The proof is clear from the definition of an inclusive Information retrieval function T, for a_i and $a_{i+1} \subseteq R$ as:

$a_i \geq a_{i+1} \rightarrow T(a_i) \subseteq T(a_{i+1})$

which confirms the previous statement that more are the terms added in the query, the fewer documents are retrieved and also vice versa. If there is a least element 0 in R, and if each document of D is retrievable, then T(0) = D which is shown as follows. For each query r$\in$R, and $r \geq 0$, hence $T(r) \subseteq T(0)$. If all documents are retrievable, then for each d$\in$ D, there is a query r $\in$R so that d is in T(r). Hence T(0) = D. Hence in an inclusive system, everything is included if no term is specified in the query. Also what

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

happens when new requests are made based upon old requests? For example, suppose there are two requests r, s $\in$ R, and the new requests r$\wedge$s and r$\vee$s are the glb and lub of r and s in R. The lub query r$\vee$s contains all the terms from r and also from s and hence it has a larger term set and hence the retrieved document set should be smaller i.e. $T(r \vee s) \subseteq T(r)$ and $T(r \vee s) \subseteq T(s)$. Also similarly, the glb query r$\wedge$s contains only the common terms between queries r and s. Hence the term list is shorter in the glb (the query is more general) and hence the retrieved document list should be larger i.e. $T(r \wedge s) \supseteq T(r)$ and $T(r \wedge s) \supseteq T(s)$. The glb and lub queries are shown in fig 61 graphically.

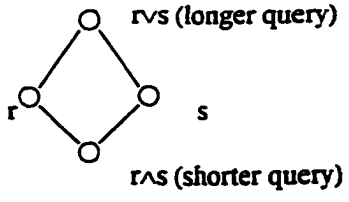


Figure 61:Operations in Query space

Theorem 2. T is an inclusive retrieval function if and only if

$$T(r \vee s) \subseteq T(r) \wedge T(s) \subseteq T(r) \cap T(s)$$

and

$$T(r \wedge s) \supseteq T(r) \vee T(s) \supseteq T(r) \cup T(s)$$

where r $\vee$s and r$\wedge$s exist and $\wedge$ and $\vee$ are the glb and lub operations in the partial order document set $L=\{T(r): r \in R\}$, also $\cup$ and $\cap$ are the set union and intersection operations in the space of document subsets that are formed by the images of queries under T.

Proof: This proof is again evident from the inclusion properties of the information retrieval system as well as the definition of glb and lub of partial ordered sets.

It is known that,

$$r \vee s \geq r \;\; ; \;\; r \vee s \geq s$$

and

$$r \wedge s \leq r \;\; ; \;\; r \wedge s \leq s$$

Since T is an inclusive function, so

$$T(r \vee s) \subseteq T(r) \;\; ; \;\; T(r \vee s) \subseteq T(s)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and

$T(r \wedge s) \supseteq T(r)$; $T(r \wedge s) \supseteq T(s)$

Hence by the definition of glb and lub.

$T(r \wedge s) \supseteq T(r) \vee T(s)$

and

$T(r \vee s) \subseteq T(r) \wedge T(s)$

Furthermore as both $T(r)$ and $T(s) \subseteq 2^D$, so the glb and lub are synonymous with $\cap$ and $\cup$. Hence it is proven that

$T(r \vee s) \subseteq T(r) \wedge T(s) \subseteq T(r) \cap T(s)$

$T(r \wedge s) \supseteq T(r) \vee T(s) \supseteq T(r) \cup T(s)$

Thus the two lattices one for the query space and one for the document images under function T are displayed graphically in fig 62.

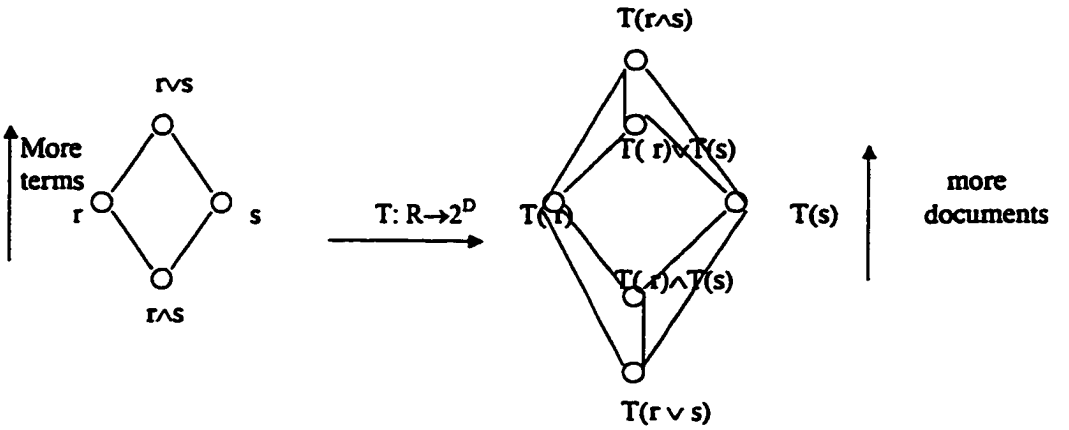


Figure 62:Request-document mapping for combined requests

Conversely, if $T(r \vee s) \subseteq T(r) \cap T(s)$, whenever $r \vee s$ is defined, then $r \geq s$ means $r \vee s$ is defined and is r. So $T(r \vee s) = T(r) \subseteq T(r) \cap T(s) \subseteq T(s)$. This means $T(s) \supseteq T(r)$., thus showing T is inclusive.

2.18.1.1 Classification of Documents

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

There are certain systems where the retrieval function does not produce the documents in the image, but instead it produces some document descriptors. The document descriptors are basically a set of terms or concepts etc. . The set of permissible document descriptors C may not be the same as the set of all queries. For example, the set of document descriptors may be a set of keywords as X, Y, Z etc. ordered by

set inclusion into a lattice as in fig 60 whereas the query space can be more complex in structure (than just set inclusion relation) to accommodate a query as " get document with topic common to terms X and Y". A typical situation of such type is given in fig 63 for three terms X, Y, Z where these terms may be juxtaposed in the document description space whereas in the query space the set intersect operation may be allowed.

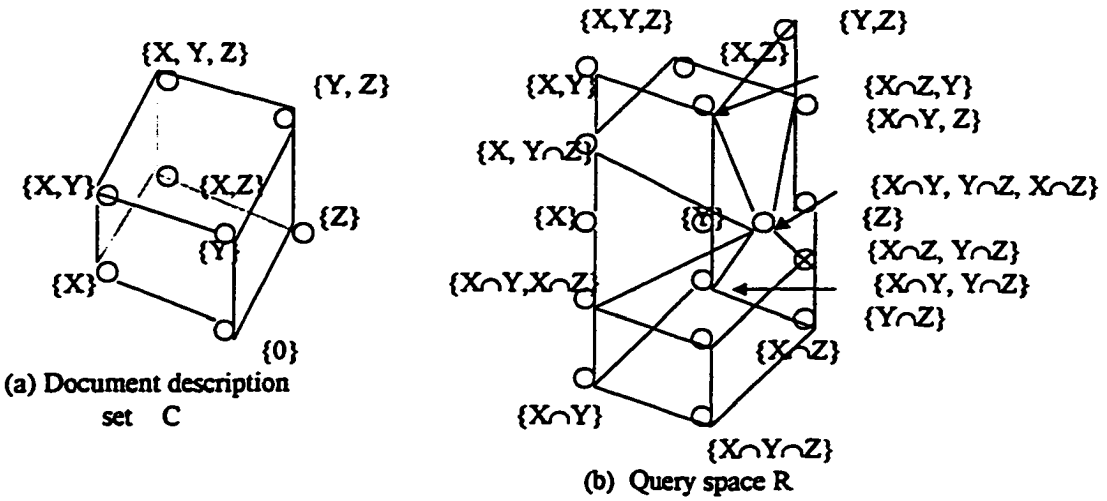


Figure 63:Description set C and query space R.

{X,Y} means topic dealing with X as well as Y. But {X∩Y} means topic common to X and Y. Consider a function X from the space D of documents into the space C, the set of all possible document descriptions. X may be called the classification function, and the value of X(d) is called the document description for document d ∈ D. The mapping X: D→C is the classification function which defines an equivalence relation on D, where a given document d_i is called equivalent to another document d_j, if and only if $X(d_i) = X(d_j)$. An information retrieval system is said to be based upon a classification if there exists a classification language C and a classification function X: D→C assigning each document d ∈ D a document description X(d) such that, for each request r∈R, T(r) is a union of equivalent classes in D

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

under the equivalence relation defined by X. As an example, X: D→C may be a function that assigns a unique integer to d ∈ D in which case the equivalence class of d is {d}.

Now, given the query space R and a document description space C, it is possible to define a new function F mapping from space R to space C.

$F: R \rightarrow 2^C$ such that for each query $r \in R$, $F(r)$ is the collection of document descriptions corresponding to the equivalence classes in D such that $T(r)$ is the union of the same equivalence class. That means if

$F(r) = \{X(d): d \in T(r)\}$ for each $r \in R$ then

$T(r) = \{d: X(d) \in F(r)\}$

A complete information retrieval system based on classification may now be taken as a request language R, a document set D, and a classification language C, together with the functions $X:D \rightarrow C$ and $F:R \rightarrow 2^C$, which define the retrieval function $T: R \rightarrow 2^D$ by

$T(r) = \{d: X(d) \in F(r)\}$

Thus F maps queries into collections of possible relevant document descriptions and the retrieval function T retrieves the documents corresponding to those collections of document descriptions. This complete model is in fig 64.

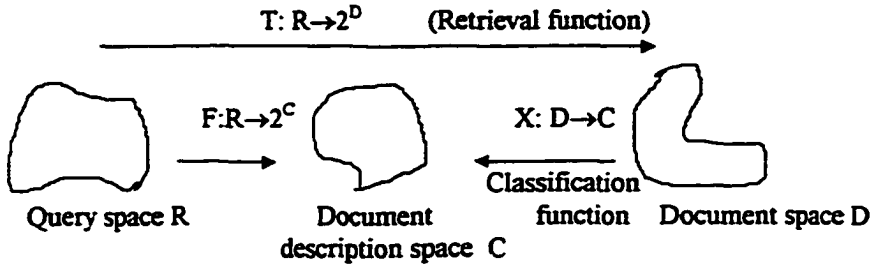


Figure 64:Retrieval model using document descriptions

Both the models as in fig 59 and fig 64 represent standard keyword search systems where a query is expressed as a set of keywords which is transformed into a set of document descriptions, usually identified

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

by the same keywords set; all documents marked with these descriptions are then retrieved. These models can be slightly extended so that the retrieved documents are ordered decreasingly by the probability of relevance. A cutoff value is chosen and all documents with request-document correlation greater than the cutoff are retrieved.

It is needed to analyze the behavior of the retrieval system with some perturbation in the query. In a viable system, if the query is slightly altered, the change in the retrieved set should also be small. In theorem 2 it was shown for an inclusive retrieval system $T(r \vee s) \subseteq T(r) \cap T(s)$ and $T(r \wedge s) \supseteq T(r) \cup T(s)$. Now under certain conditions the equality even holds.

Theorem 3. In a partially ordered retrieval system based on classification with R=C, and where $T(r) = \{d: r \leq X(d)\}$, the following relationships will hold wherever $r \vee s$ exists:

$T(r \vee s) = T(r) \wedge T(s) = T(r) \cap T(s)$

This means that the system operates in such a way that when the user gives a few keywords for search, the system will not only retrieve documents indexed by any of the keywords but also those indexed by more refined set of keywords containing the original set as a subset. Thus if the document description set C of fig 63(a) is used (which according to the theorem is identical with R), a typical query r={X} will retrieve documents indexed by keyword X, as well as documents indexed by {X,Y}, {X,Z} and {X,Y,Z}.

Proof: Step 1. T is inclusive, i.e. for any r and s in R, $r \geq s$ implies $T(r) \subseteq T(s)$.

This is true because for any $d \in T(r)$, X(d) >= r which means X(d) >= s (because r >= s). So d is included in T(s). Hence $T(r) \subseteq T(s)$. So T is inclusive.

Step 2. Now applying theorem 2 it can be said,

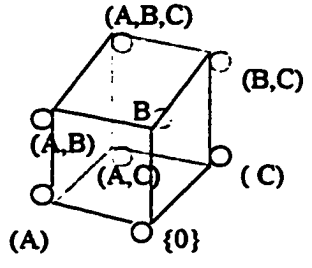
$T(r \vee s) \subseteq T(r) \cap T(s)$ Furthermore, for any $d \in D$ in $T(r) \cap T(s)$, $d \in T(r)$ and $d \in T(s)$. Hence, r <= X(d) and s <= X(d). So by definition of lub, $(r \vee s) <= X(d)$. So d must be included in $T(r \vee s)$. Thus $T(r) \cap T(s) \subseteq T(r \vee s)$. Combining these two predicates leads to $T(r \vee s) = T(r) \cap T(s)$ - the result of the theorem.

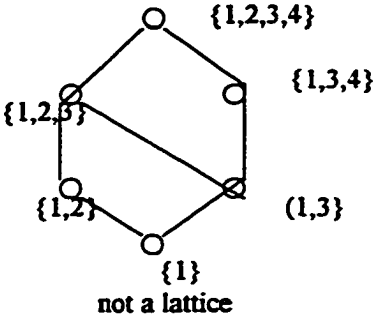
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

An example can be given to illustrate the result. Fig 65 a shows four documents 1, 2, 3, 4 are being indexed by 4 sets of keywords as {B}, {B,C}, {A, C}, {A,B,C} and the document description space is the same as query space (R). Fig 65(a) also gives the full mapping of X and T functions. In the query, any combination of terms A, B, C are allowed. The query space and the retrieved document space are in fig 65(b) and fig 65(c). Now consider an example query as (A,C) ∨ (B) = {A,B,C}.

D	X →	C = R	T →	T(R)
-		(0)		(1,2,3,4)
-		(A)		(1,2)
4		(B)		(1,3,4)
-		(C)		(1,2,3)
-		(A,B)		(1)
2		(A,C)		(1,2)
3		(B,C)		(1,3)
1		(A,B,C)		(1)

(a) retrieval mapping from different spaces





not a lattice

(b) Request space R ○ indicates document description (c) retrieved document space (T(R))

Figure 65:Example retrieval system

T((A,C) ∨ (B)) = T(A,B,C) = {1}

T(A,C) = {1,2} T(B) = {1,3,4} So, T(A,C) ∩T(B) = {1}= T((A,C) ∨ (B)) which confirms the result of the theorem i.e. T(AC∨B) = T(AC)∩T(B). Also, note that equality does not hold true on the inverse of it (on T(r∧s)). But under certain condition the equality holds in both ways as in the following theorem.

Theorem 4 In a retrieval system based on classification with R = C and T(r) = {d: r ≤ X(d)}, if R is a lattice, and T is a one-to-one mapping from R into 2^D, then

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$T(r \vee s) = T(r) \cap T(s)$

and

$T(r \wedge s) = T(r) \cup T(s)$ where $\cup, \cap$ are the lub and glb in the image set T(R) of documents under the mapping T.

Proof: By theorem 3, T is inclusive (see step 1) and hence $T(r \vee s) = T(r) \cap T(s)$. Also using theorem 2, $T(r \wedge s) \supseteq T(r) \cup T(s)$. So the goal is to prove that $T(r \wedge s) = T(r) \cup T(s)$.

Hence,

$T(r \wedge s) \supseteq T(r)$ (2.17)

$T(r \wedge s) \supseteq T(s)$ (2.18)

Now assume any document set T(b) in the image set T(r), such that $T(b) \supseteq T(r)$ and $T(b) \supseteq T(s)$. As R is a lattice, so $r \vee b$ and $s \vee b$ exist. Now $T(r \vee b) = T(r)$ and $T(s \vee b) = T(s)$. As this mapping is one-to-one, so $r \vee b = r$ and $s \vee b = s$. That means, $r \geq b$ and $s \geq b$. So, $r \wedge s \geq b$. So $T(r \wedge s) \subseteq T(b)$. Now one obtains $T(r \wedge s)$ which is a superset of T(r) and also of T(s) by eqn 2.17 and 2.18. Also for any b, where $T(b) \supseteq T(r)$ and $T(b) \supseteq T(s)$, T(b) contains $T(r \wedge s)$ and $T(r \wedge s) \supseteq T(r) \cup T(s)$; this is the definition of an lub for T(r) and T(s) and $T(r \wedge s)$ is the lub of them. So

$T(r \wedge s) = \text{lub}(T(r) \text{ and } T(s)) = T(r) \cup T(s)$ - QED.

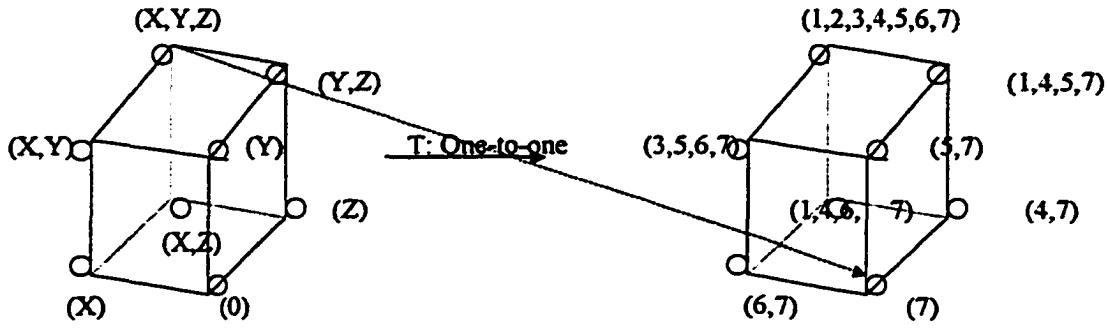
Consider, as an example, the retrieval mapping of fig 66.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

D	X	C = R	T	T (R)
0		{0}		{1,2,3,4,5,6,7}
1		{X}		{1,4,5,7}
2		{Y}		{2,4,6,7}
3		{Z}		{3,5,6,7}
4		{X,Y}		{4,7}
5		{X,Z}		{5,7}
6		{Y,Z}		{6,7}
7		{X,Y,Z}		{7}

Document space | document description space | retrieved document space

(a) Sample one-to-one retrieval mapping



(b) Query space R

(c) space of retrieved documents T(R)

Figure 66:Example retrieval model - one-to-one mapping

Consider a request as (A,B ∧ A,C) in the example.

T(A,B ∧ A,C) = T(A) = {1,4,5,7}

T(A,B) = {4,7}

T(A,C) = {5,7}

So, T(A,B) ∪T(A,C) = {1,4,5,7} in T(R) = T(A,B ∧ A,C)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

It may be noticed that the image space of fig 66 is not the same as the space 2^D, which has 128 points and where $\{4,7\} \cup \{5,7\} = \{4,5,7\}$. But this point does not exist in the space T(R). But the nearest upper bound within the eight points is {1,4,5,7} in T(R).

The conditions specified for theorem 4 are too stringent, although the theorem stands true even where the T mapping is not one-to-one. The mapping does not need to be one-to-one if, in the original assignment of the description sets to the documents, a given description c_i is assigned to a specific document d_i only when all possible description sets c containing c_i, are also assigned to some documents by the mapping $X:D \rightarrow C$[1].

When the query space is not same as the description space, the model described before can still apply if a suitable function is defined between R and C. For example, if the query space is as in fig 63 (b) and the document description space is as in fig 63(a), then each request r can be broken into many requests r_i so that for each r_i the original theorems apply.

2.18.1.2 Use of complementary

So far the information identifiers have been perceived as to stand for positive properties of the documents such that no internal relations exist among the various identifiers. A given identifier is assigned to a document or a query when the property applies to the corresponding documents and if does not apply then it is not assigned. Using this understanding, one can use set intersection, set union or even complement of sets of documents. The complement of the set P of documents identified with identifier X is the set of documents $\bar{P}$ to which identifier X is not assigned. The usual complementation rule holds here

$$P \cap \bar{P} = \emptyset \quad \text{and}$$

$$P \cup \bar{P} = I$$

[1] In that case, a sub-lattice does however exist in R for which the conditions for theorem 4 are satisfied.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In many systems, it is useful to work with negative information identifiers where for example, $\bar{P}$ means *not blue* if P means a property *blue*. This can have two interpretations:

1. The negative property is considered to be another independent property like other existing properties, such that when one says *not modest* , he is looking for a document identifier which is '*not modest*' or immodest. So this is not a complement but a property. At this time, one does not need to know if there can be another identifier as '*modest*'.

2. The negative property is a logical complement of a positive property and each document is identified by either presence or absence (positive or complement) of each identifier. For example, in the vector representation a 0 will mean a negation and a positive value will mean presence of a concept. Here the set complementation $\bar{X}$ can be used and also the properties specified above hold true. For example,

   $X \cup \bar{X} = Y \cup \bar{Y}$ and $X \cap \bar{X} = Y \cap \bar{Y}$ (2-19)

There are some slight difficulties with the complementation where negative identifiers are used in the following conditions:

1. If the collection of documents is dynamic (due to addition and deletion of documents) with time, then the complement operation is not so well defined and thus $X \cup \bar{X}$ is not the same as $Y \cup \bar{Y}$.

2. Now there is a distinction between labels *blue or not blue* and *modest or not modest* because the first predicate refers to the concept of color (so the object of description is physical) and the second predicate refers to the concept of character (there has to be a character involved with the object of description) and the documents may not be the same. Thus eqn 2.19 may not apply because there will always be items that are neither in X nor in $\bar{X}$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In these circumstances the set-theoretical boolean operations are no more applicable, since a well-defined set complementation is not defined. Also when the documents collection is dynamic, some items are continually added and some are deleted. In such a growing system the set-theoretical model may apply in certain subsections of the collection, but not in general because there are items not yet marked but still are generally accessible to the user. Thus the system can not access "all those and only those items" that are called for. Instead, a search can be restricted to the labeled items only to retrieve "only those but not all those" items because some of the wanted items may still be unmarked. Alternatively, one can show all the current holdings and also all unmarked items thus retrieving "all those and not only those" which is a low-precision and high-recall result. In such cases, two types of complements exist for a set marked X.

1. X^*, called the pseudo complement, which is the largest set of items that certainly contains no Xs. Thus X^* contains all items that are *not X* but contains no unmarked items although some of them may be Xs (but they are unknown). The double pseudo complement X^{**} represents the smallest set in the collection containing certainly Xs.

2. $\neg X$, the Browerian complement, that is the smallest set of items which certainly contains all the not Xs. This set has all the unknown documents, since some may be not X 's and the double complement $\neg\neg X$ is the largest set which certainly contains nothing but X's.

The difference between the two types of complements is illustrated graphically in the following fig 67. It has been suggested that either one of the two algebras, that is either pseudo-complemented or Brouwerian algebra can be used as the retrieval model in a dynamic environment. In these algebras the positive element is replaced by the double complement. The pseudo-complemented algebra can be used to operate with a high-recall performance since X^{**} is the smallest set certainly containing all the Xs. But the Brouwerian algebra can be used for high precision operation because $\neg\neg X$ is the largest set containing only the Xs.

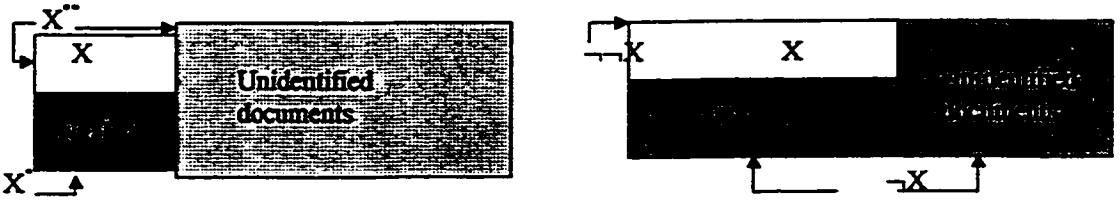
Uncertainty in $X = X^{**} - \neg\neg X$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Thus all the set operations using set complementation can be carried out using one of the two versions, either pseudo-complement or Brouwerian complement.

Another approach to deal with the unknown documents is topological space. Many ideas have been offered to use topological space of open sets superimposed on the original set structure for retrieval models. Neighborhoods are defined and unknown items are assigned to these neighborhoods which they appear to be closely related to. In general, any neighborhood of a term is interpreted as a set of terms that is conceptually similar or relevant, and the collection of neighborhoods for a topological space T is assumed to provide a representation of the relations between elements of T. The neighborhoods themselves are defined for each element as open sets containing the specified element. As before, it is possible to use the algebra of closed sets of a topological space for high precision searches and the algebra of open sets (i.e. with the document neighborhood) for high-recall searches.

In addition, it appears that the concept of metric space is also involved in document retrieval. The normal set-theoretical model can be used within the metric space under the assumption that the set complements are defined. Consider a set X and let d be the transformation from $X \times X$ into the field of real numbers. Then d is a real-valued function defined over the set of all ordered pairs of elements of X and d is the metric in X if the following criteria are met:



(a) Pseudo-complement (b) Brouwerian complement

Figure 67:Complementation of collections

1. $0 <= d(m, n) < \infty$
2. $d(m, n) = 0 \Leftrightarrow m = n$
3. $d(m, n) = d(n, m)$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4. $d(m, n) + d(n, o) \geq d(m, o)$

Here m, n, o are arbitrary elements of set X.

A metric matching function d is well suited for use in information indexing and retrieval , because the metric allows sufficient structure in the index space to identify groups of related documents. By the dint of the fourth property - a search request that is close to a document m, must be close to the other documents that are close to m. A set X with the ,metric d defines a metric space {X, d} where d(m, n) is the metric distance between element m and n from set X.

If T is a set of terms or documents, T can then be assigned a metric by defining a distance measure as $d_0 = 1$ when $m \neq n$, and a measure $d_0 = 0$ when $m = n$. It can be shown that each subset of a metric space $M = [X,d_0]$ is open so that metric d_0 defines a topology in X. These sets consist of null set and all possible subsets of X including X itself. These sets form a Boolean algebra, and the retrieval models based on Boolean algebras are special cases of topological theories.

2.18.1.3 Tree and graph structures

If the information system is defined using the identifiers sets and no additional structures are being defined except for the inclusion relations, then the set-theoretic model works fine. Previously it was seen that inclusion relations can be defined to lead to a partial order relations among the subsets and even lattices. But in many cases, more relations may need to be defined and then the new model take one of the two principal forms.

1. If a single type of relation is defined between the elements of the set of documents and the relation is like an ordering relation, in general it can be represented by a tree structure where each element has only one parent (except the root), all the elements are bound to somebody in the relation and some nodes are related to no one. Such relations lead to tree structures. Library classification system, semantic similarity relations in a thesaurus are some examples.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2. If many relations are present. the nodes on the many trees (each for a relation) again get interconnected thus forming an abstract graph.

Syntol graphs are an example of attempts to use abstract graph model in information retrieval. It is advantageous to store more relations between the entries of the dictionaries. Such can be drawn to have a term correspond to a node and a relation between two terms shown by an edge. While building the graphs, some important tasks are to be identified.

1. Groups of highly related terms for which paths exist in the graph between each node and other nodes needs to be identified; Such groups of vertices in a strongly connected cycle can become a synonym class.

2. Dictionary look-up operations must be carried out, for example, by path tracing methods which will be the corresponding ones for tree matching methods (as discussed before). So algorithms will be needed as something like tree tracing methods.

3. Sometimes two or more documents each having its own graph may have to be compared by graph matching processes and their algorithms are required.

4. When dictionaries are replaced by graphs, algorithms must be written on adding and deleting entries I the graph.

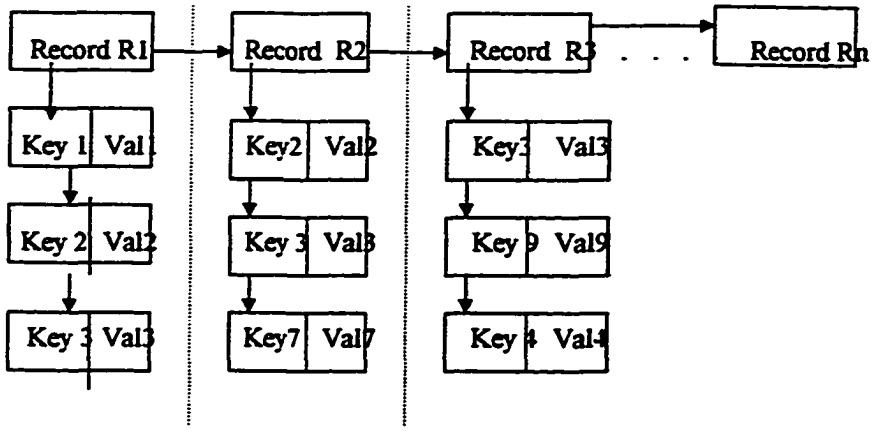
Such graphs may also be used to represent relations between documents. For a very large-sized graph with many nodes and edges, the above operations are plentiful and far too complex. These graph models can be used better in representing the structure of information files, made of records for the documents including both key and data. Examples of the file structures in the record-oriented way are given in fig 68 where file is built in the order of record (or document number). The fig 69 provides the file-structures for key-oriented storage. The fig 68 (a) provides a high-level record layout. Each record has a record-header made of record number, record length, record type etc. Then it has all the keys (in an ordered list) that are applicable to the document and their values. Then the record has the document

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

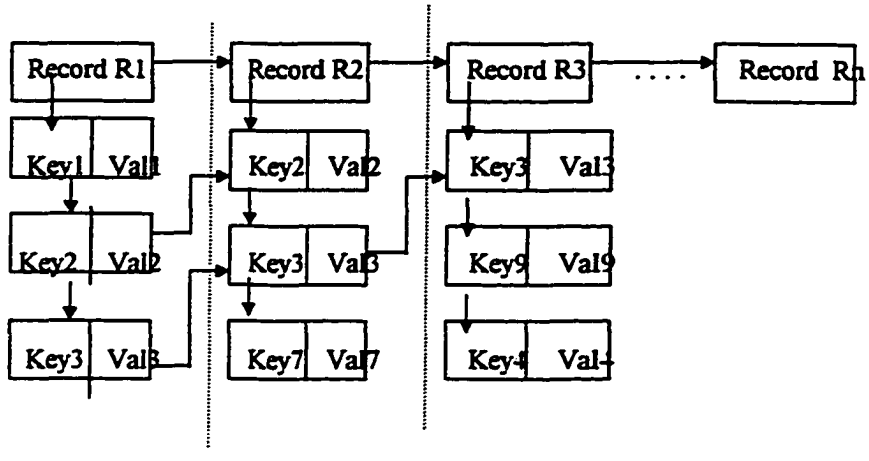
information as bibliographic information, abstract (and may be a pointer to the text). The fig 68 (b), each record is linked to the next record, each key is linked to the next key. This results in a tree structure. In addition, one may want the identical keys (or identical values within the same keys) to be linked across many records also. This gives the file structure as in fig 68 (c) which looks like a graph rather than a tree. Then it is easy to search documents identified by a key *Key1*. The key-oriented structures are in fig 69 where they are arranged by the keyword order. Here the subsequent keywords are linked in a chain. In addition, all the records that have a common keyword are also linked together as shown in fig 69(b). The record structure of fig 69(a) has record header information made of key length, no of document-records in the current record etc.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(a) basic record structure



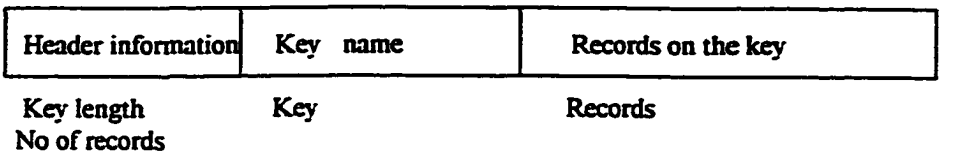
(b) Record-oriented file structure in a 2-dimensional chain



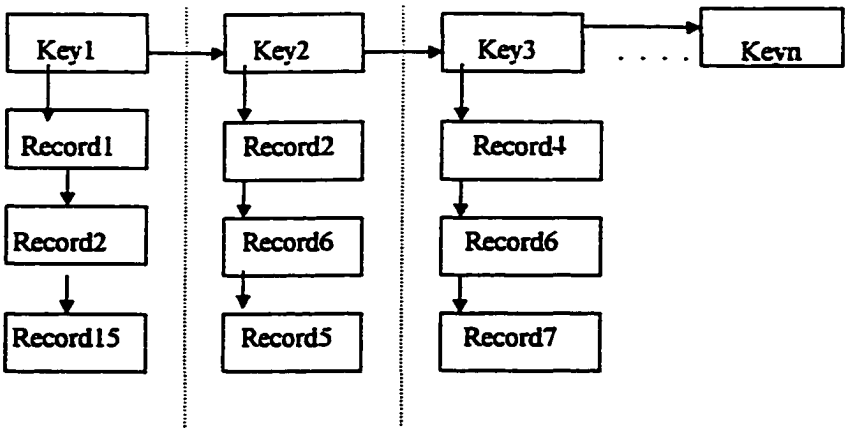
(c) 3-D chain : The extra chain is between the same keywords across multiple records

Figure 68:File structures for record-oriented storage

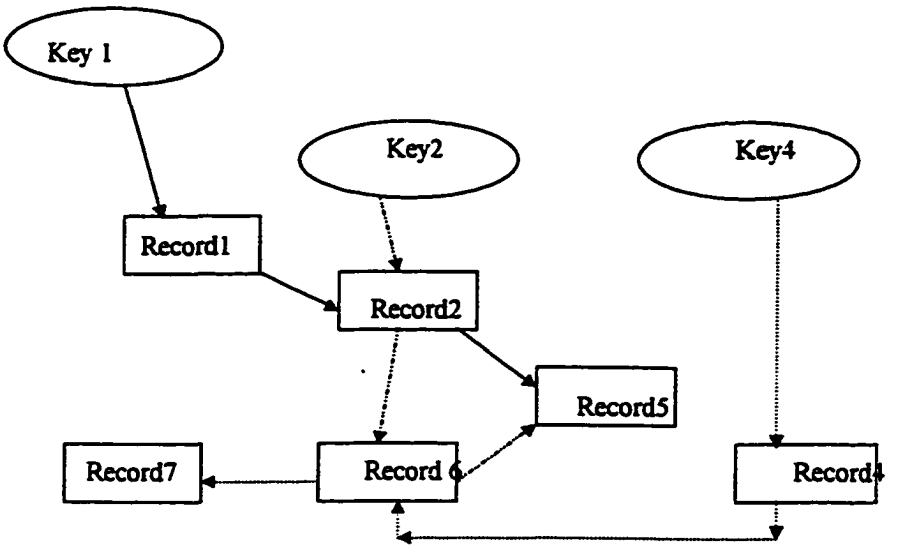
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



(a) basic record layout



(b) Key-oriented arrangements : 2-D chain



(c) Key-oriented multi-list file structure (3-D chain)

Figure 69:Key-oriented file structures

In the case of fig 68c, it is easily possible to retrieve the records that is identified with a given key say key1 or a given value of the key *key1* say val1. In fig 69b, a single chain is maintained under

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

each key and the result structure is a tree. If it is decided to add to the single chain under each key a second chain linking identical entries for different keys then a graph structure as in fig 69c results. Here only one occurrence of the record is being shared by multiple chains. Such structures are called multi-list system; the key-oriented files are also sometimes called inverted files.

2.19 Correlation formulae for document vectors

As important as document indexing and storage is the concept of query searching and retrieval. As the part of a search, a query vector has to be correlated with the document vector. Such correlation measures may be based on various criteria as the number of matching vector elements or the weights for the elements or total no of concepts in a vector etc. Sometimes the vectors are binary where weights are 0 or 1 meaning presence or absence of a property. But in most cases, they are real values where high value indicates a strong property. One normalized formulae (value between 0 and 1) for computing correlation between vectors X and Y is:

$$r_{XY} = \frac{\Sigma_{i=1}^{t} X_i . Y_i}{\Sigma_{i=1}^{t} X_i + \Sigma_{i=1}^{t} Y_i - \Sigma_{i=1}^{t} X_i . Y_i} \qquad (2\text{-}20)$$

Another normalized function used widely is cosine angle function which gives some kind of cosine of the angle subtended by the two vectors X and Y. So,

$$r_{XY} = \frac{\Sigma_{i=1}^{t} X_i . Y_i}{\sqrt{\Sigma_{i=1}^{t} (X_i)^2 . \Sigma_{i=1}^{t} (Y_i)^2}} \qquad (2\text{-}21)$$

But when very long vectors are used, this function returns a small value because the length of the vector comes in the denominator. The normalized *overlap* function below in eqn 2-22 does not have such

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

limitation because the minimum of the length of either two is used in the denominator. This overlap function is a very reasonable function giving good results on many operations.

$$r_{XY} = \frac{\sum_{i=1}^{t} \min (X_i, Y_i)}{\min(\sum_{i=1}^{t} X_i, \sum_{i=1}^{t} Y_i)} \qquad (2\text{-}22)$$

Also there are asymmetric versions of the above correlation function as:

$$r_{XY} = \frac{\sum_{i=1}^{t} \min (X_i, Y_i)}{\sum_{i=1}^{t} X_i} \qquad (2\text{-}23)$$

$$r_{XY} = \frac{\sum_{i=1}^{t} \min (X_i, Y_i)}{\sum_{i=1}^{t} Y_i} \qquad (2\text{-}24)$$

Another binary function suggested by Maron and Kuhns is based upon the concept of excess values of the matching terms over the non-matching terms above the total of co-occurrence and mismatches. This function is also commonly used.

$$r_{XY} = \frac{(\sum_{i=1}^{t} X_i . Y_i) (\sum_{i=1}^{t} \bar{X}_i . \bar{Y}_i) - (\sum_{i=1}^{t} X_i . \bar{Y}_i) (\sum_{i=1}^{t} \bar{X}_i . Y_i)}{(\sum_{i=1}^{t} X_i . Y_i) (\sum_{i=1}^{t} \bar{X}_i . \bar{Y}_i) + (\sum_{i=1}^{t} X_i . \bar{Y}_i) (\sum_{i=1}^{t} \bar{X}_i . Y_i)} \qquad (2\text{-}25)$$

This formula may be written in terms of frequencies of occurrences.

$$r_{XY} = \frac{F_{XY} . F_{\bar{X}\bar{Y}} - F_{X\bar{Y}} . F_{\bar{X}Y}}{F_{XY} . F_{\bar{X}\bar{Y}} + F_{X\bar{Y}} . F_{\bar{X}Y}} \qquad (2\text{-}26)$$

Consider two specific documents m and n that exist in the document collection. There are now 4 types of properties for document m and n. Properties that can be jointly assigned to documents m and n are written as mn and their total number is F_{mn}. Similarly say $F_{m\bar{n}}$ is the number of properties which are present in document m and absent in document n, $F_{\bar{m}\bar{n}}$ is the number of properties which are absent in document m as well as in document n. If S was the total count of properties (cardinality of the vectors), then

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$S = F_{mn} + F_{m\bar{n}} + F_{\bar{m}n} + F_{\bar{m}\bar{n}}$ (2-27)

Also $F_m = F_{mn} + F_{m\bar{n}}$ and $F_n = F_{mn} + F_{\bar{m}n}$ (2-28)

If the concepts are independent of each other, then the probability of a concept being jointly assigned to document m and document n is the product of the probability of the concept being assigned to document m and the probability for it being assigned to doc n. So,

$P(m, n) = P(m) \cdot P(n)$ (2-29)

This is

$$\frac{F_{mn}}{S} = \frac{F_m}{S} \cdot \frac{F_n}{S}$$

So, $$F_{mn} = \frac{F_m \cdot F_n}{S}$$ (2-30)

Here the excess of the joint assignment above the estimated value can be a measure of correlation value. To watch the strength of the correlation, one can watch the difference between F_{mn} and the product of eqn 2-30. So $\delta_{mn} = F_{mn} - (F_m \cdot F_n)/S$ (2-31)

Now, the following criteria may be used in the construction of the correlation function as:

1. The correlation r_{mn} between document m and n is 0 when $\delta_{mn}=0$ and r_{mn} should vary proportionately with δ_{mn}.
2. The maximum correlation $(r_{mn})_{max}$ is achieved when the property vector for document m is contained within the vector for document n or vice versa; that is each property assigned to m is also assigned to j (i.e. $F_{i\bar{j}}=0$) or vice versa ($F_{\bar{i}j} = 0$) or when both vectors are identical (i.e. $F_{i\bar{j}} =0$ and $F_{\bar{i}j}=0$).
3. The minimum correlation $(r_{mn})_{min}$ occurs when the property vector of document m is contained in the complement of the property vector for document n (i.e. $F_m = F_{m\bar{n}}$ which means $F_{mn} = 0$) or vector j is contained in the complement of vector i (or $F_{ij} = 0$) or, one property vector is the complement of the other ($F_{i\bar{j}}$ and $F_{\bar{i}j}=0$).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4. The correlation measure should vary between -1 and +1.

This correlation measure in eqn 2-26 varies between -1 and +1 and its numerator is equivalent to $S.\delta_{mn}$

$$S.\delta_{mn} = S\left(F_{mn} - \frac{F_m.F_n}{S}\right)$$

$$= S F_{mn} - (F_{mn} + F_{m\bar{n}})(F_{mn} + F_{\bar{m}n})$$
$$= S F_{mn} - F_{mn}(F_{mn} + F_{m\bar{n}} + F_{\bar{m}n}) - F_{m\bar{n}} . F_{\bar{m}n}$$
$$= S F_{mn} - F_{mn}(S - F_{\bar{m}\bar{n}}) - F_{m\bar{n}} . F_{\bar{m}n}$$
$$= F_{mn} . F_{\bar{m}\bar{n}} - F_{m\bar{n}} . F_{\bar{m}n}$$

Thus eqn 2-26 satisfies all the above criteria of a correlation function.

Another correlation function is based on the magnitude of the deviation from independence of the joint property assignment over and above the assignment of individual properties to the documents. Also it is based on the total no of documents in the collection. The formula is given as:

$$r_{XY} = \frac{n(n+1)}{2}\left(\frac{\Sigma_{i=1}^{t} X_i . Y_i}{t} - \frac{\Sigma_{i=1}^{t} X_i . \Sigma_{i=1}^{t} Y_i}{2t^2}\right) \qquad (2\text{-}32)$$

If binary vectors are used and the property frequencies are used, then the formula becomes

$$r_{XY} = \frac{n(n+1)}{2}\left(\frac{F_{XY}}{S} - \frac{F_X . F_Y}{2S^2}\right) \qquad (2\text{-}33)$$

From these two above equations, one can say

$$\delta_{XY} = F_{XY}/S - F_X.F_Y/S^2.$$

It is clear here, that the above two formulae are dependent upon S which depends on the total no of documents in the collection. As more documents are added, S increases and so the joint correlation changes. If such correlation factors are used to calculate inter-document correlation which may be needed for indexing, the indexing scheme may need to change because of large changes in S. So, such ever-changing formulae may not be suitable for large use.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The overlap and cosine functions and the function in eqn 2-26 are possibly the most commonly used correlation functions. Further improvements in the choice of correlation functions may be of secondary importance compared to more important needs in the area of document analysis.

2.20 File structures

Today the query-search processing has turned towards on-line processing although previously it used to be batch processing. There is one most important file that contains the document information. Two types of file organizations are mainly used to store the keywords, their weights, bibliographic data etc. as *direct file* and *inverted file*. In the direct file system, the file is stored in order by document numbers, and as documents are added, they are assigned higher numbers to be added sequentially at the end of the file (the document records are all added at the end). The document record contains all the keywords, their weights, the bibliographic data etc. . The direct files are very useful in batch processing but because the files are sequential, the online response-time will be slow.

The second type of file organization is *inverted file structure* where the data is sorted by the information identifiers or the keywords. Thus all documents indexed by keyword A come first, then all documents indexed by keyword B and then for keyword X etc.. The keywords can be the indexes and the file organization can be based upon a B-tree. The indexes can be saved in a B-tree and the leaf nodes of the B-tree can point to the pages of a flat file that contains the series of document numbers. The following fig 70 illustrates an exemplary file organization for an inverted file.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

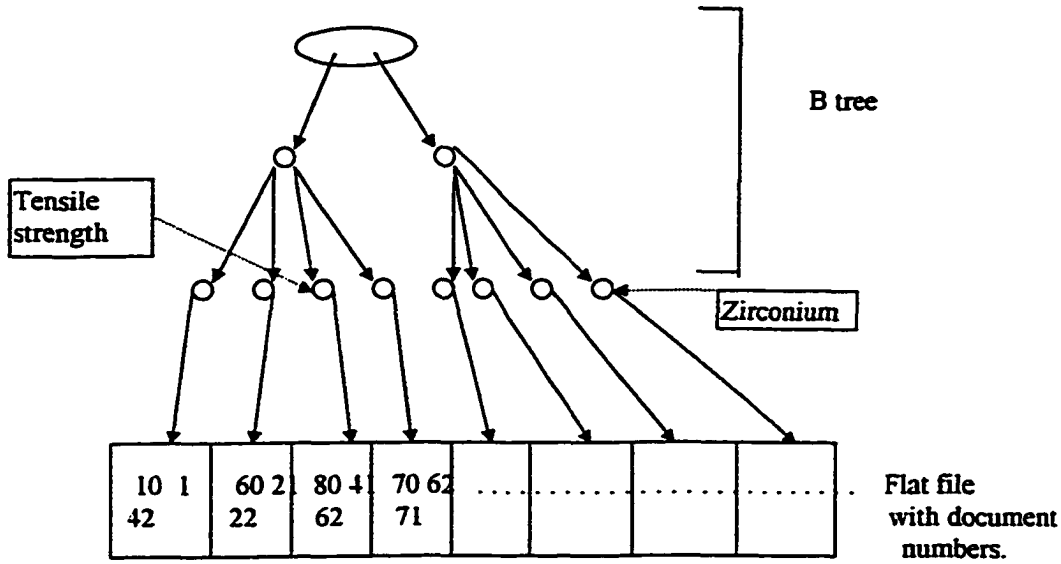


Figure 70:Inverted file organization using B-tree

To retrieve information stored in an inverted file, one has to access the pages of the flat file for the given keyword. The keyword has to be searched within the B-tree and then the respective pages from the flat file can be accessed using the pointer value of the corresponding leaf node. But in pure inverted file, the pointer on the leaf node points directly to the corresponding document records. But if it is a combined file system, then as the next step of the search, the document (which may exist on a random access file) must be accessed to verify if the search logic is being satisfied. Here the document records can be stored in an indexed file (indexed by the document number). The following fig 71 shows the flow-diagram on the major steps to be followed for building the inverted file including the key directory. The inverted file is normally built on a random access device where the speed of access does not depend on the current position of the reading head.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

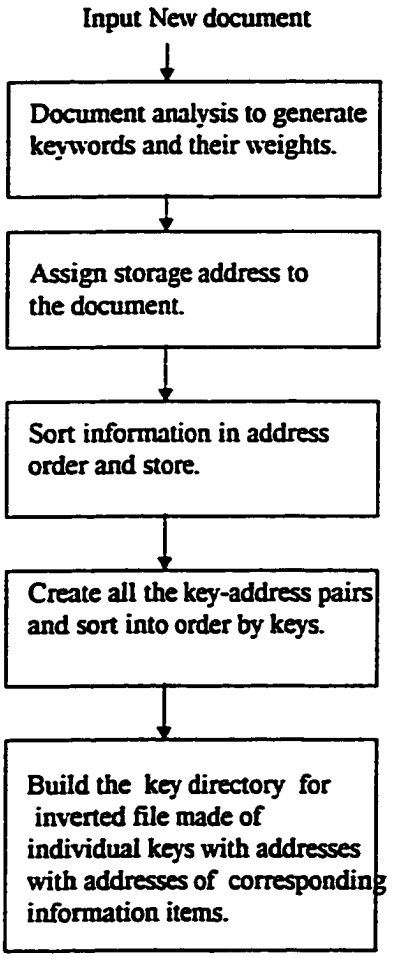


Figure 71:Building inverted file including key directory

The key directory is the set of page numbers pointed to by the leaf nodes of the B-tree (or it can be an alternate data structure as a 2-3 tree etc.). The flat file contains a series of pages or buckets containing all the document pointers (like document numbers etc.) for the given keyword. In a simple scheme, each page can have a page link on its top and then the data part which can contain all the document numbers. In case the page overflows, another page may be assigned and the two pages may be linked. This is like overflow processing in a hashed algorithm for data storage. The details of a page structure is shown in fig 72 below.

Again, this is only the picture of a combined file system. In the pure inverted file system these pages contain the document records. But in a combined system, this is a list of document numbers (or document

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

addresses). Such a combined system is better, because there are searches involved at these level also. If these pages contain short document numbers, then searches can be faster, even update, deletion etc. happens faster. So it is a better idea to gain efficiency of operation at the cost some extra small amount of storage for storing the document addresses instead. The updates, deletions happen faster because it leaves the document record as it is in the document file, but only updates or deletes the address of the documents which is faster.

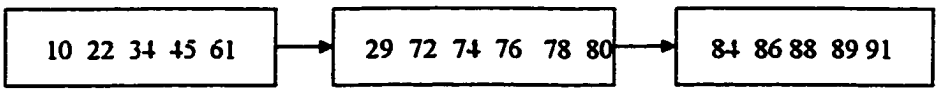


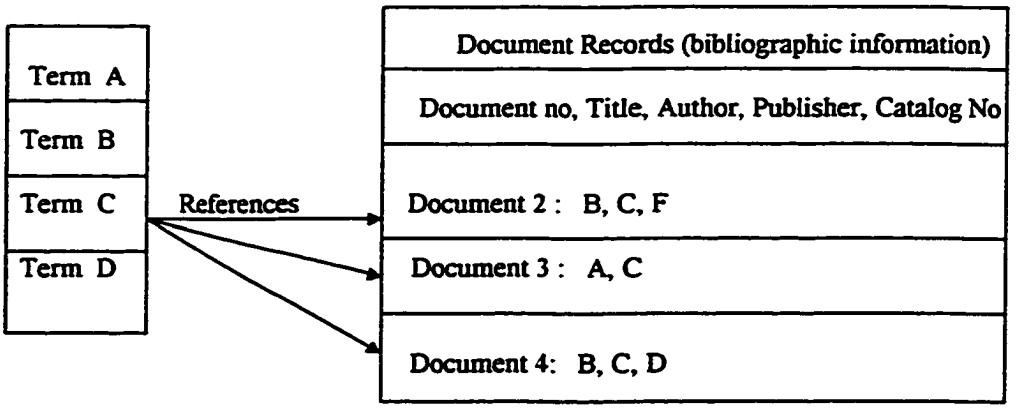
Figure 72:Pages of the inverted file with 2 overflow pages (combined file system)

Fig 71 describes a simple flow-chart for the initial construction of an inverted file. First the page assignment for the new information is handled first and then the key directory is constructed. The steps are as: The input document is analyzed first to generate the list of keywords and their weights. Assign storage area to the input data and record the address. Now create the key-address pairs and sort them in order of keys. Now construct the key directory for inverted file made of individual keys with addresses of the corresponding documents.

Retrieval using inverted files (i.e. using the look-up method) is normally more efficient than having to search all the documents in a sequential way. In this way, the search begins with searching the key directory which is maintained in a highly suitable data structure to support a fast search and if the search succeeds it results in a pointer to the document record in the document file. If the key directory is implemented using a data structure as in fig 70, then once the keyword is found in the B-tree as a single node, a pointer will be leading to the first page that contains a list of document addresses (or document numbers) corresponding to the given keyword. Each page has two areas. First, the page link which may be bi-directional to link to other pages with the same keyword. The second part is the data part of the page

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

which contains a series of words of document numbers (each word a document number) or document addresses. When a page overflow occurs, a new page is allocated and the new page is linked to the old page through the page link. The relationship between the inverted index file and the direct document file is shown in fig 73 below.



(a) Inverted index file (b) Direct file with document records with bibliographic information

Figure 73:Reference relations in a combined file system

The multi-list system:

In the multi-list system, the storage efficiency in the reduced key directory comes from the one entry per key policy. But this storage efficiency is gained at the cost of added search time since the entries under each key is not directly accessible, but are chained. The construction method is described below in fig 74.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

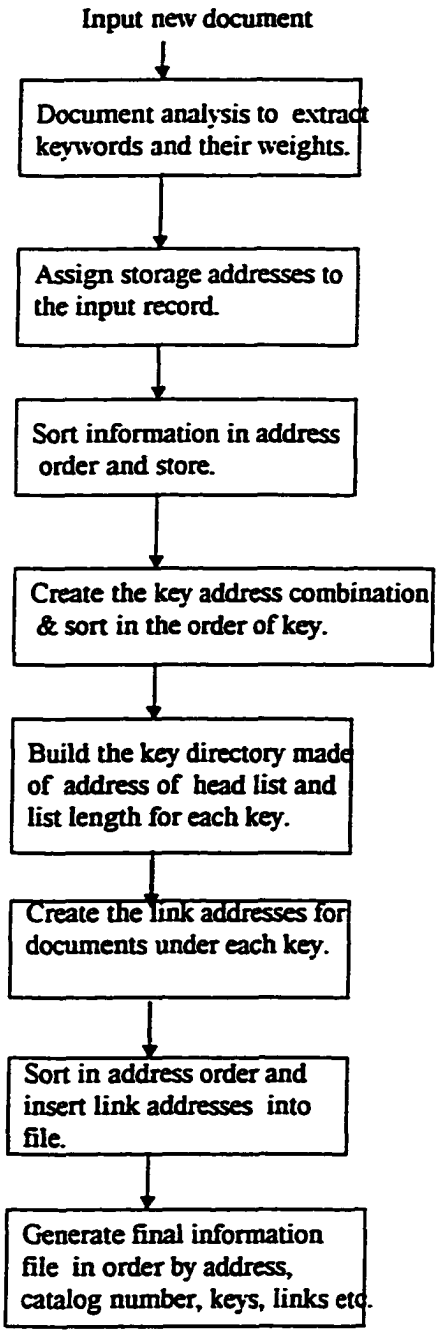


Figure 74:Flow-chart for building a multi-list with key directory

Storage addresses are first assigned to the documents, and the data are sorted in address order and stored. The key address information is created, and the small key directory is built up made of for each key, the address of the head list and the number of entries in the list. That means for example, if a key participates

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

in 3 lists (equivalently 3 graphs), then the key entry will point to a list of 3 entries each of which will point to the respective list (or graph). An example is shown in fig 75 where node K_1 participates in 3 graphs (with 3 colors of the edges). So there are 3 lists to be connected at this node. The way it is implemented is the following. There is a list of 3 nodes (actually one more for the head node) and each node points to a list (i.e. the graph) by pointing to the next element of the list.

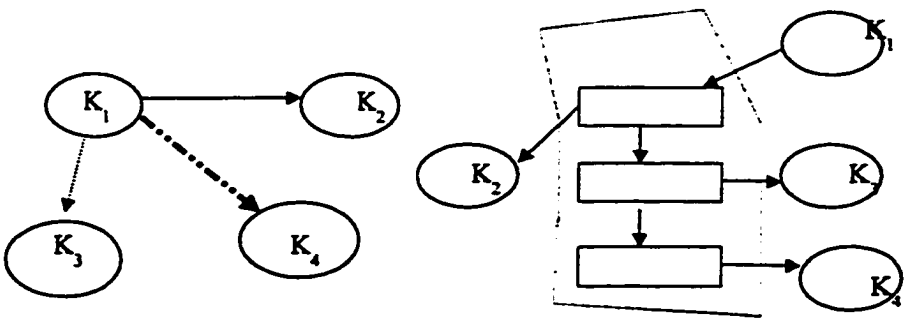


Figure 75:A close picture of a multi-list (node K1 participates in 3 lists)

So corresponding to node K_1, two things need to be created as the node and a linked list surrounded by the dotted line. Again continuing the line of creation of the multi-list, after the key-address combination is created, such pairs are arranged in the key order (sets up the directory). Now the list pointers (surrounded by dashed line) have to be generated for each key and inserted into the file after they are sorted in the address order.

The multi-level searching

In systems with very large collections (of the order of millions of documents), choosing a file system becomes a very difficult issue. A direct file system requires a full search of the entire file and thus fails to return a reasonable response time. Simultaneously, an inverted file does not perform very well, because the search may be less restrictive than searching over a few keywords. For example, it could be a search on a given subject and some wild characters in the keywords. One approach to searching such large collection is to follow some classification method that would restrict the search to a small subset. Using some document clustering algorithms which compares the properties of documents, it is possible to

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

form document groups or clusters such that the correlation between each pair within the group is larger than the correlation of documents in different groups. Once such clustering is done, one can construct a centroid vector which works as a representative classification vector for the cluster. The concepts on the centroid vector represent the properties of the cluster. One simple method is calculating a vector average of all the member documents. Thus, c is defined as:

$c = (1/n) \Sigma_{i=1}^{n} (d_i)/|d_i|$ (2-34)

where the class consists of $D=\{d_1, d_2, d_3, ...,d_n\}$. Here d_i is the i-th document vector and n is the size of the class. Alternatively, the centroid vector c may be a unit vector with an angular orientation for which the $\Sigma_{i=1}^{n}[\theta(c, d_i)]^2$ is minimized where θ is the angle of orientation between two t-dimensional vectors c and d_i of concepts and $-180^\circ \leq \theta \leq 180^\circ$. In the geometrical representation of the document space, each document vector $d_i / |d_i|$ is a unit vector terminating on a unit t-sphere (because $x^2 + y^2 + z^2 + ... = 1$ is a t-dimensional unit sphere).

Once the centroid vectors are available for each of the clusters, it is easy to specify a two-level algorithm as follows:

1. A new search request is first checked in conjunction with the centroid vectors of each of the m given clusters.
2. The query vector is compared with document vectors of only those classes whose centroids exhibit a high correlation with the query.

There are m clusters and say N documents. So on average each cluster has about N/m documents. Say i cluster centroid vectors had matched the query. So total number of comparisons is

$C = m + i.\ N/m$ (2-35).

This is minimized when $1 - i.\ N/m^2 = 0$ or $m = \sqrt{i.N}$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and $C_{min} = 2\sqrt{i.N}$ (2-36)

which is much better than N searches.

As the number of documents per cluster (which is given by N/m) is constant across the document space, so in the region where document density is very high, the cluster size is small and the cluster size is large where document density is low. In a system with a large number of document clusters, further economy is achieved through maintaining another level of clustering on top of the document clusters. Such a three-level clustering leads to higher savings in search time (order of $N^{1/3}$) if documents are uniformly distributed with concepts. There is a classification done on the centroid clusters. The search proceeds as follows:

1. The classification vectors for all the centroid clusters are matched with query vector.
2. Next step, the centroid vectors in the matching clusters are again compared.
3. Finally, the individual documents corresponding to highly matching centroid vectors are individually compared.

With the experiments it has been found that nothing but a full search will avail if very high recall is required. Recall certainly decreases by 15-20% using clusters compared to full searches; for average recall levels, a two or three-level cluster organization, needing only one-fifth of the searches of full searches results in little loss in precision. For low recall levels, the precision of a two-level cluster search is about 5-15 % smaller than that of a full search. A standard multi-level search is effective depending on if the queries are really compatible with the structure of the document clusters. Or are rather far contrary to the classification vectors of the centroid clusters. Thus if a query falls neatly in the center of a document cluster, a large correlation is found when such queries are matched with the centroid vectors. On the other hand, if the query vector is not included in any document clusters or it falls in the periphery of a document cluster, then the match coefficients with all the centroid vectors will be weak and so detail document search must be pursued. Fig 76 below shows such weak queries (the triangles represent queries and the squares represent document vectors and the large polygons represent the clusters) in a clustered

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

document space. If such queries are coming in commonly and the queries are similar among themselves with minor variations then forming query clusters may be very effective. The query clustering algorithm basically may continue along the following steps:

1. The previously entered queries are clustered into groups using a standard clustering method.
2. An associated document subset is formed for each subset of queries constructed in step 1; the associated subset of documents includes all documents that are highly correlated with the query subset. This process is carried out for all query subsets.
3. All documents not associated with any query cluster by step 2 are divided into subsets, using a standard clustering method.

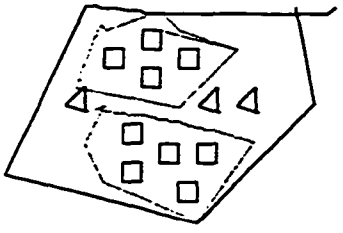


Figure 76:Odd queries in a clustered document space

Correspondingly, the search algorithm also will be modified as:

1. A new query will be first matched with the document clusters of the previous queries; if anything matches then the document vectors within the matching clusters will be compared with the document vectors.
2. If nothing matches in step 1, then the query is matched in conjunction with the centroid vectors of the non-associated document-clusters. If anything matches, then the detailed documents are compared.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

This results in more efficient search than multi-level search, because fewer subsets have to be compared with the query vector.

2.21 Iterative search using Relevance feedback

One drawback that people may complain about the information retrieval systems is that the retrieval system does not pay attention to the subjective needs of the user. After all each user is different and the system is not attentive to the subjective needs of the user. Each user-entered query is handled independently and matched with all the document vectors. There are schemes to rectify such problems by rebuilding the query from the original query by adding extra terms that may be more tuned to the user needs. The revised queries may be constructed by the user herself or may be built automatically using the historical information. If the reconstruction is manual, the system provides an automatic dictionary (possibly in another window) supplying the users dictionary information about the terms used in the query and also suggesting alternate rewordings for query reformulation. The automatic dictionary can be constructed in many ways: One, simply for each term in the original query, the synonyms, the hierarchically related concepts and the related terms may be displayed on the screen for user's attention. The user can click or copy and paste any term she likes.

The automated reformulation of query may be following many procedures, and one of them may be relevance feedback. method. This method progresses in the following way: It starts an initial search using the original query and presents part of the result in the decreasing order of relevance. Of these, the user selects some to be relevant (by accessing them and spending more time) and some non-relevant for his purpose. These relevant documents are reported to the system whose properties act as model and system adds some of the additional related properties of those documents in the original query. The altered query is answered and the results are presented to the user. This process can be continued iteratively. A few properties are added at a time, if correlations between the relevant documents (found as relevant before) and the modified query increase, then the progress is retained and the query is answered. Again a part of the retrieved set is presented to the user and user's entered new relevance assessments can be

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

further used for the next finer reformulation etc.. A simple flow-chart for the simple relevance feedback system is given in fig 77 below.

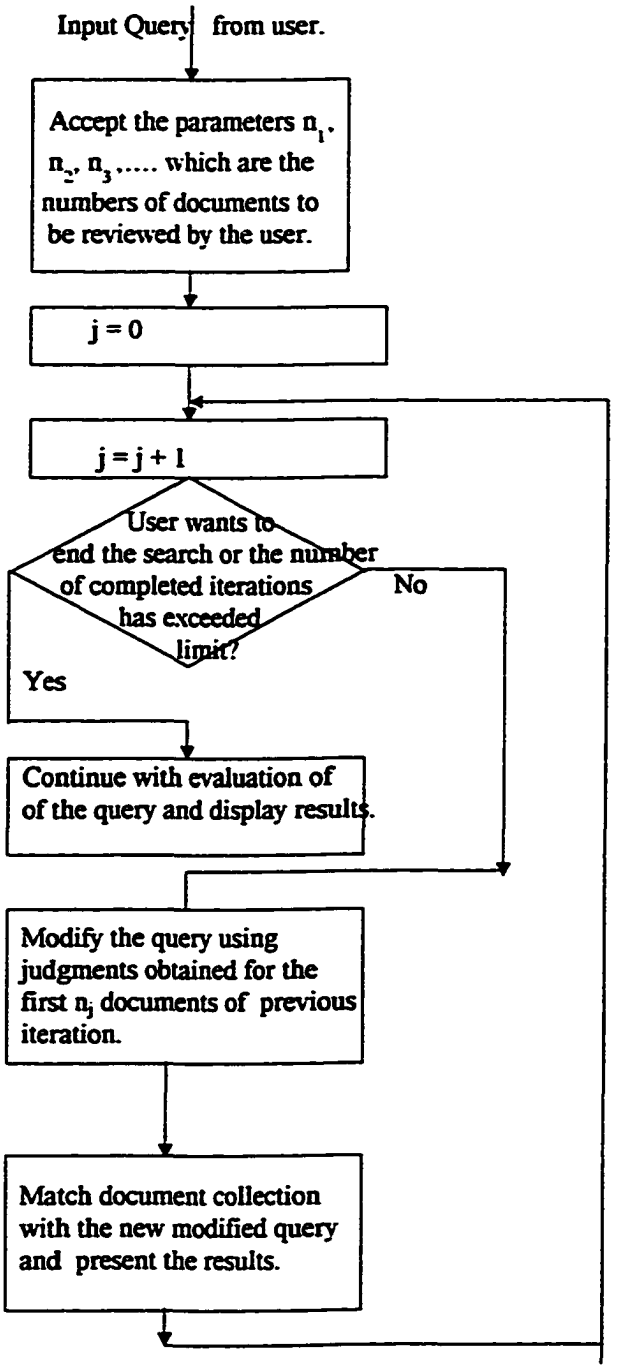


Figure 77:Flow-chart for a simple relevance feedback system

This flow-chart shows how the relevant terms are promoted and non-relevant terms are demoted and additional terms keep being added. The method followed at each stage of query reformulation is the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

formula that tries to maximize the difference in request-document correlation between relevant and non-relevant document sets. Specifically, if D_R is the non-empty document subset termed as relevant, then an optimal query is the one that maximizes the distance between the subset D_R from the rest of the collection ($D - D_R$). In other words, if c(q, d) is the correlation function (or distance) calculating the correlation between query q and document d, then an optimal query q_0 is the query that maximizes the distance function F as given by:

$$F = \bar{c}_{d \in D_R}(q,d) - \bar{c}_{d \notin D_R}(q,d) \qquad (2\text{-}37)$$

where $\bar{c}$ is the average distance function between query q and all relevant documents (i.e. $d \in D_R$) and less distance means higher query-document correlation.. Now the relevant set of documents D_R is built up incrementally through user's interaction. Instead of producing optimal query q_0 directly, it is generated incrementally through the iterations and starting with an initial query q_0 that identifies a part of D_R. As more subsets of relevant documents are added to D_R, the sequence of modified queries slowly approach the optimal query q_0. For an efficient method, the number of iterations should be small and the rate of convergence is reflected in the stability of the retrieved set.

One query reformulation scheme may be outlined here as an example although there are many more of such kind. This scheme tries to produce an optimal query q_0 that differentiates the partial set of relevant documents identified by the user from the remaining documents. Assuming N documents in all of which n_0 are identified as relevant (i.e. $n_0 = n(D_R)$), the function F of eqn 2-37 can be rewritten as:

$$F = \frac{1}{n_0} \sum_{d \in D_R} \frac{q \cdot d}{|q|\,|d|} - \frac{1}{N - n_0} \sum_{d \notin D_R} \frac{q \cdot d}{|q|\,|d|}$$

$$= \frac{q}{|q|} \left[\frac{1}{n_0} \sum_{d \in D_R} \frac{d}{|d|} - \frac{1}{N - n_0} \cdot \sum_{d \notin D_R} \frac{d}{|d|} \right] \qquad (2\text{-}38)$$

This equation is in the form of an ordinary vector product as $F = q \cdot x$. The optimal query q_0 maximizes F which is proportional to x and so,

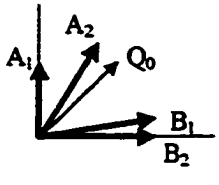
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$q_0 = k\left(\frac{1}{n_0}\sum_{d \in D_R}\frac{d}{|d|} - \frac{1}{N-n_0}\sum_{d \notin D_R}\frac{d}{|d|}\right) \qquad (2\text{-}39)$$

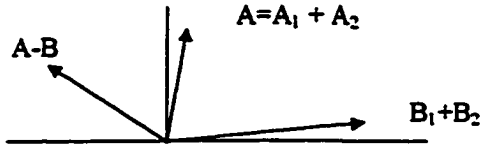
The query reformulation can be written as:

$$q_{i+1} = n_1 n_2 q_i + n_2 \sum_{i=1}^{n1}\frac{r_i}{|r_i|} - n_1 \sum_{i=1}^{n2}\frac{S_i}{|S_i|} \qquad (2\text{-}40)$$

where q_i is the i-th query of a sequence, and $R = \{r_1, r_2, r_3, \ldots, r_{n1}\}$ is the set of relevant document vectors retrieved in response to query q_i . Here the sets S and R provide the feedback after the i-th iteration. A 2-D graphical illustration of eqn 2-40 is given below in fig 78 which include the document vectors and the query vectors. Fig 78a shows the original query vector (Q_0) and four document vectors (A_1, A_2, B_1, B_2). Of these four documents, A_1 and A_2 are relevant documents and B_1 and B_2 are non-relevant documents as marked by the user. Fig 78b shows the modifier vector A - B. The resultant normalized query vector Q_1 is created from the original query vector and the modifier vector as a sum of q_0 and the modifier vector (A - B). It may be verified that whereas the original query Q_0 is oriented approximately midway between relevant and non-relevant documents, the modified query Q_1 is tuned more towards the relevant documents as A_1 and A_2 and away from the non-relevant documents. So such a query will certainly improve the relevance or the precision.

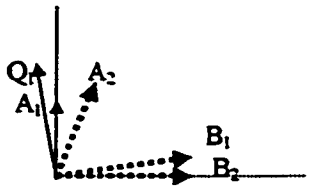


(a) Original query Q_0 and relevant docs (A_1, A_2) and non-relevant docs (B_1, B_2)



(b) Sum of the relevant docs minus the non-relevant docs.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



(c) Resultant query Q_1 as a sum of original query Q_0 and modifier.

Figure 78:Graphical illustration of relevance feedback

The recall-precision effect of relevance feedback has been verified in further details which proves the improvement in both precision as well as the recall.

2.22 Assessment of information retrieval systems

Quality of the performance of an information retrieval system means its ability to satisfy the user which depends on various internal factors within the system. When discussing the question of systems evaluation, the following factors determine the quality of an information retrieval system. These are:
Collection: The coverage of the collection, types of documents available, reliability of the abstract etc..

Indexing: Type of indexers, level and accuracy of the indexers, depth of indexing required, complexity of indexing language etc..

Query Analysis & search: Type of searching, power & complexity of search mechanism, search time required, accuracy of search, clustering methods etc..

Operating efficiency: Cost considerations, service problems, time lag and response time.

The quality of the system depends on the following major criteria:

1. The coverage of the collection. The extent to which it can satisfy the user.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2. Response time or the average time gap between query and its answer.
3. The type of presentation on the output
4. Amount of search efforts, computer cost, network cost etc..
5. Recall of the system which gives fraction of the relevant material that is retrieved.
6. Precision of the system that gives fraction of the retrieved material that is relevant.

The measures of recall and precision

To measure the recall and precision of an information retrieval system, the documents have to separated into four categories depending on their relevance and retrieved status as in fig 79 where cardinalities of the categories are called a, b, c, d .

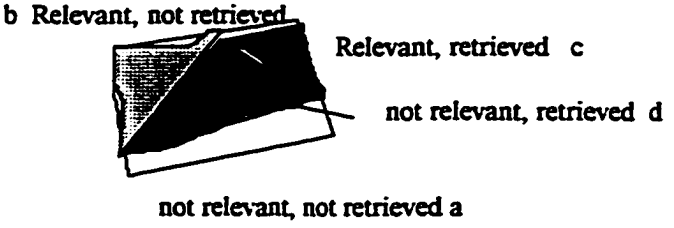


Figure 79:Four categories of documents

Using these definitions (as given before), one can say

$$\text{relevance} = \frac{c}{b+c} \qquad (2\text{-}41)$$

$$\text{precision} = \frac{c}{c+d} \qquad (2\text{-}42)$$

Of course, the concept of relevance is subjective and so can change from person to person, but here it is presupposed that they can be resolved. So relevance and precision can be measured for all documents with respect to all queries.

In fig 80, there are two graphs showing recall measures of a system along the y-axis and the ranks of the retrieved documents along the x-axis. Such a graph starts at zero and stays at zero until a

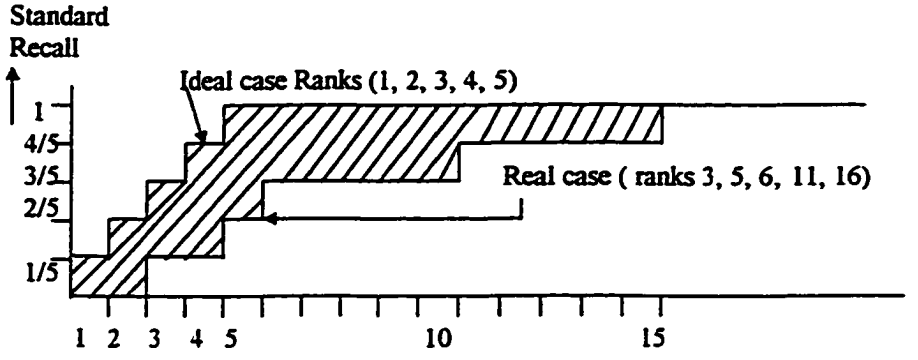
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

rank is reached corresponding to a relevant document, where the recall jumps to 1/n, if there are n relevant documents in the system. The recall then stays at 1/n until the next relevant document rank is reached, at which point the recall increases to 2/n, and so on, until the last relevant document rank is reached; then the recall reaches a final value of n/n or 1. If this recall function is plotted on the same graph for an ideal system where the n relevant documents are ranked as 1, 2, 3, ..., n then the area contained within these two functions can be termed as a measure of the recall of the system. Actually more the area, lower the recall. An example real case is shown in fig 80a in a system of 25 documents of which 5 are judged relevant (n=5). The ranks of these documents are say 3, 5, 6, 11 and 16.

It is easy to show that the area between the ideal system curve and the real system recall function curve (hatched area) is given by

$$\frac{\sum_{i=1}^{n} r_i - \sum_{i=1}^{n} i}{n} \qquad (2\text{-}43)$$

where n is the number of relevant documents for the query in the system and r_i are their ranks.

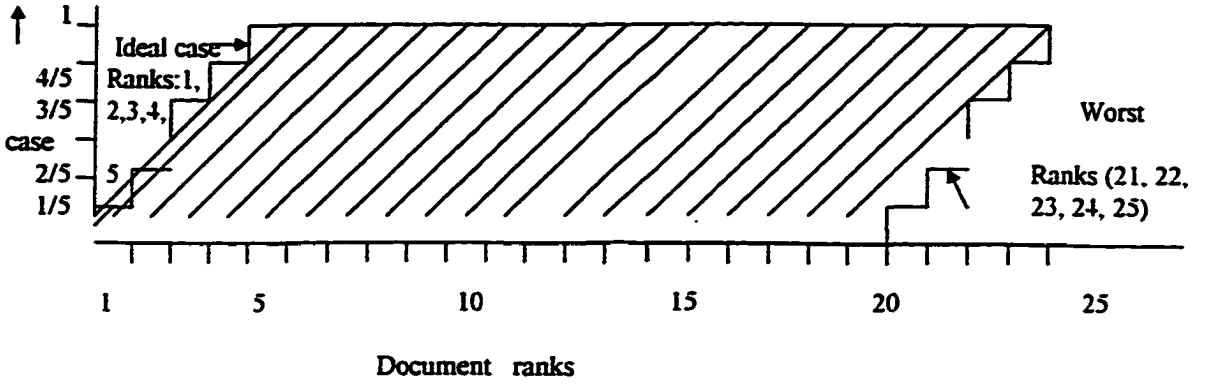


(a) Recall differences for a real case (r_i = 3, 5, 6, 11, 16)

$$\frac{\sum_{i=1}^{n} r_i - \sum_{i=1}^{n} i}{n} = \frac{26}{5} = 5.2$$

Standard recall

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



(b) Recall differences for worst case (r_i=21,22,23,24,25)

$$\frac{\sum_{i=1}^{n} r_i - \sum_{i=1}^{n} i}{n} = N-n = 20$$

Figure 80:Graphical illustration of recall measure

Consider the area under the curve of ideal systems. This area underneath the curve is given by

$$A_{ideal} = \sum_{i=1}^{n-1} i/n + \sum_{i=n}^{r_n} n/n$$

where the first term is the area under the initial step function and the second term is the area under the curve once the value 1 is reached.

Now the area under the real curve is given by

$$A_{real} = \sum_{i=1}^{n-1} (i/n)(r_{i+1} - r_i)$$

The area between the two curves is then

$$\begin{aligned} A_{ideal} - A_{real} &= \sum_{i=1}^{n-1} i/n + \sum_{i=n}^{r_n} n/n - \sum_{i=1}^{n-1} (i/n)(r_{i+1} - r_i) \\ &= (1/n)\,[\, n(r_n - n) - \sum_{i=1}^{n-1} i\,(r_{i+1} - r_i - 1)\,] \\ &= (1/n)\,[n(r_n - n) + \sum_{i=1}^{n-1} (r_i + i) - (n-1)\,r_n] \end{aligned}$$

By expanding the sum and collecting terms, the difference can be reduced to

$$A_i - A_t = (1/n) \sum_{i=1}^{n} (r_i - i)$$

which is given by eqn 2-43. In the example this area difference is 26/5.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

This above function varies from 0 (for perfect retrieval where retrieval ranks are 1, 2, 3, 4...,n) to value N-n for the worst possible case as in fig 80(b) where the document ranks are N-n+1, N-n+2, ...,N. This function may be normalized so that it exists between 0 and 1 by dividing by N-n; subtraction from 1 finally ensures that the measure equals 1 for the best case and 0 for the worst case. This normalized recall is given as:

$$R_{norm} = 1 - \frac{\sum_{i=1}^{n} r_i - \sum_{i=1}^{n} i}{n(N-n)} \qquad (2\text{-}44)$$

A normalized precision measure can be derived similarly using a graphical illustration and by computing the area difference between precision curve for ideal case and precision curve for an actual system. The hatched area in fig 81 gives a measure of the precision. Consider the area under a typical precision curve. The area between the two relevant documents of rank r_i and r_{i+1} is clearly

$$\sum_{r_i \le r \le r_{i+1}} \frac{i}{r} = i \sum_{r_i \le r \le r_{i+1}} \frac{1}{r}$$

for $1 \le i < n$, since I relevant documents will have been retrieved at that point. Anyway, for i=n there is no document of rank r_{i+1}; hence the area under the precision curve beyond r_n is

$$n \sum_{r_n \le r \le N} \frac{1}{r}$$

The total area under the precision curve is given by:

$$A = \sum_{i=1}^{n-1} \left(i \sum_{r_i \le r < r_{i+1}} \frac{1}{r} \right) + n \sum_{r_n \le r < N} \frac{1}{r}$$

$$= \sum_{r_1 \le r < r_2} \frac{1}{r} + 2 \sum_{r_2 \le r < r_3} \frac{1}{r} + \ldots + n \sum_{r_n \le r < N} \frac{1}{r}$$

The expression may be rewritten as follows:

$$A = \left(\sum_{r_1 \le r < r_2} \frac{1}{r} + \sum_{r_2 \le r < r_3} \frac{1}{r} + \ldots + \sum_{r_n \le r < N} \frac{1}{r} \right)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$+ \left(\sum_{r_2 \le r \le r_3} \frac{1}{r} + \ldots + \sum_{r_n \le r \le N} \frac{1}{r}\right)$$

$$+ \left(\sum_{r_3 \le r \le r_4} \frac{1}{r} + \ldots + \sum_{r_n \le r \le N} \frac{1}{r}\right) + \ldots + \left(\sum_{r_n \le r \le N} \frac{1}{r}\right)$$

$$= \left(\sum_{r_1 \le r \le N} \frac{1}{r}\right) + \left(\sum_{r_2 \le r \le N} \frac{1}{r}\right) + \ldots + \left(\sum_{r_n \le r \le N} \frac{1}{r}\right)$$

$$= \sum_{i=1}^{n} \left(\sum_{r=r_i}^{N} \frac{1}{r}\right)$$

Standard precision

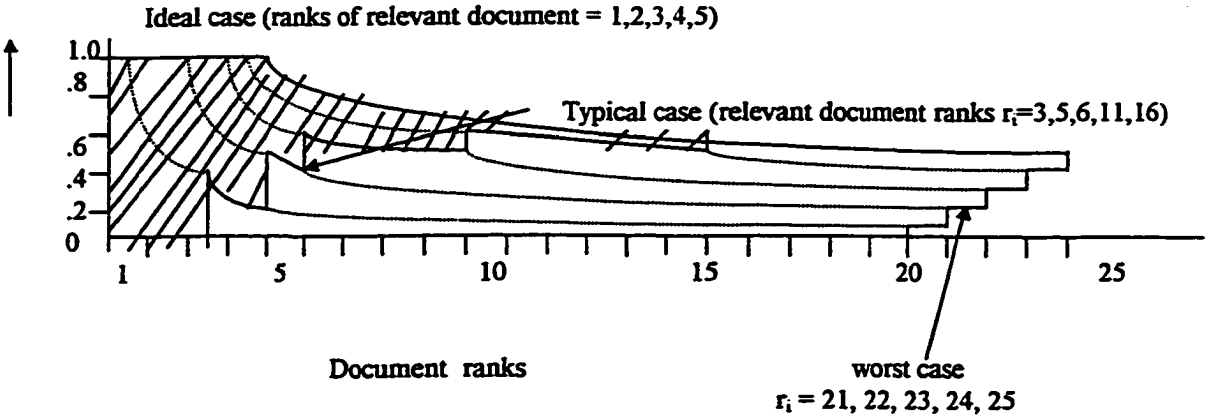


Figure 81:Construction of normalized precision measure (n=5)

Substituting the inner sum by an integral and evaluating, one gets the following result:

$$A = \sum_{i=1}^{n} \int_{r_i}^{N} 1 \cdot dr = \sum_{i=1}^{n} (\log N - \log r_i)$$

$$= n \log N - \sum_{i=1}^{n} \log r_i$$

In the ideal situation, $r_i = 1, 2, 3, \ldots, n$, and the area under the ideal curve is then

$$A_{ideal} = n \log N - \sum_{i=1}^{n} \log i$$

The difference in area between ideal and typical curves is given by:

$$A_{ideal} - A = n \log N + \sum_{i=1}^{n} \log r_i - n \log N - \sum_{i=1}^{n} \log i$$

$$= \sum_{i=1}^{n} \log r_i - \sum_{i=1}^{n} \log i \qquad (2\text{-}45)$$

This area becomes maximum in the worst case that is the n precision documents are ranked as N-n+1, N-n+2, . . . , N. The values of eqn (2-45) may be normalized to vary between 0 and 1 by considering as

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

before the maximum possible area obtained by taking the worst case where $r_i = N - n + 1, N-n+2, \ldots, N$. In that case, the area under the precision curve is

$$A_{worst} = \sum_{i=1}^{n} \log (N - n + i) - \sum_{i=1}^{n} \log I$$
$$= \sum_{i=1}^{n} \log (N-n+i) / i$$
$$= \log \prod_{i=1}^{n} (N-n+i) / i$$
$$= \log \frac{N(N-1)(N-2) \ldots (N-n+1)}{n(n-1)(n-2) \ldots 1}$$
$$= \log \frac{N!}{(N-n)!.n!} \qquad (2\text{-}46)$$

The normalized area between the precision curves is calculated by dividing the expression of eqn (2-45) by the constant of eqn (2-46) and then the best value will be 0 and worst value will be 1.Finally a subtraction from 1 will produce the measure of normalized precision such that the best precision will be 1 and worst 0.

$$\text{So, } P_{normalized} = 1 - \frac{\sum_{i=1}^{n} \log r_i - \sum_{i=1}^{n} \log i}{\log [N! / (N-n)! \, n!]} \qquad (2\text{-}47)$$

Both the normalized functions as in equation (2-45) and (2-47) reflect overall system performance. A value of 1 for either implies a value of 1 for the other; opposite of what is noticed on standard measures of recall and precision. In addition, these two normalized measures differ from each other in the weights being assigned. One may like to use expressions of recall-like and precision-like functions which are independent of cutoff count (which eqns 2-45 and 2-47 are dependent on). In that case the new normalized recall and normalized precision can be defined to be approximations of average standard recall and average standard precision (averaged over all possible cutoff levels). Hence,

$$\Re_{norm} = \frac{1}{N} \sum_{i=1}^{N} R(i) \approx R_{normalized}$$

and

$$\wp_{norm} = 1 \sum_{i=1}^{N} P(i) \approx P_{normalized}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

where N is the total number of documents in the collection, and R(i) and P(i) are the recall and precision functions (as defined before in eqn 2-45 and 2-47) after retrieving i documents.

For the normalized recall, let r_j be the rank of the j-th relevant document and let there be n relevant documents in the collection. Then one can define a function f(i, j) such that

$$f(i, j) = 1 \text{ when } r_j \le i$$
$$= 0 \text{ otherwise}$$

It is seen that R(i), the recall at the cutoff level i, is

$$R(i) = \frac{1}{n} [\text{number j such that } f(i, j) = 1]$$

$$= \frac{1}{n} \sum_{j=1}^{n} f(i, j)$$

The averaged recall for all cutoff levels is then

$$\Re_{norm} = \frac{1}{N} \sum_{i=1}^{N} R(i)$$

$$= \frac{1}{Nn} \sum_{j=1}^{n} \sum_{i=1}^{N} f(i,j)$$

$$= \frac{1}{Nn} \sum_{j=1}^{n} \left[\sum_{i=1}^{r_j-1} f(i,j) + \sum_{i=r_j}^{N} f(i, j)\right]$$

$$= \frac{1}{Nn} \sum_{j=1}^{n} [(r_j - 1)\,0 + (N - r_j + 1)]$$

$$= \frac{1}{Nn} [Nn - (\sum_{j=1}^{n} r_j) + n]$$

$$= 1 - \frac{1}{Nn} \sum_{j=1}^{n} r_j + \frac{1}{N}$$

Here, it may be observed that if n=0, $\Re_{norm} = 1$, and when n=1 and $r_1 = 1$ then $\Re_{norm} = 1$. In other cases, max $r_j \ge n$, and so $\sum_{j=1}^{n} r_j > n$ (for $n \ge 2$) and $\Re_{norm} < 1 - 1/N + 1/N = 1$. On the other hand, $\sum_{j=1}^{n} r_j \le nN$ since the rank of the last relevant document can not exceed N, so that $\Re_{norm} \ge 1 - (1/nN)\, nN + 1/N = 1/N$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

As $1/N \le \Re_{norm} < 1$, a transformation is needed if $\Re_{norm}$ is to cover the range (0,1).

Consider then a normalized recall function as

$$R_{norm} = a\,\Re_{norm} + b = a\left(1 - \frac{1}{Nn}\sum_{j=1}^{n} r_j + \frac{1}{N}\right) + b$$

In the worst case, $r_j = N - n + j$ when R_{norm} is zero. So, one gets

$$R_{norm} = 0 = a\left[1 - \frac{1}{Nn}\sum_{j=1}^{n}(N - n + j) + \frac{1}{N}\right] + b$$

$$= a\left[1 - \frac{1}{Nn}\left[n(N-n) + \frac{n(n+1)}{2}\right] + \frac{1}{N}\right] + b$$

$$= \frac{a(n+1)}{2N} + b \qquad (2\text{-}48)$$

In the best case, one has $r_j = j$. So,

$$R_{norm} = 1 = a\left(1 - \frac{1}{Nn}\sum_{j=1}^{n} j + \frac{1}{N}\right) + b$$

$$= a\left[1 - \frac{1}{Nn}\,\frac{n(n+1)}{2} + \frac{1}{N}\right] + b$$

$$= \frac{a}{nN}\left[nN - \frac{n(n+1)}{2} + n\right] + b$$

$$= \frac{a}{2N}(2N - n - 1 + 2) + b \qquad (2\text{-}49)$$

Now subtracting eqn 2-49 from eqn 2-48 and thus solving for a and b

$$1 = 1 - 0 = \frac{a}{2N}[2N - n + 1 - (n+1)]$$

$$= \frac{a}{2N}(2N - 2n) = \frac{a}{N}(N-n)$$

So, $a = \frac{N}{N-n}$ and $b = -\frac{(n+1)}{2(N-n)}$

So the new normalized recall function becomes

$$R_{norm} = 1 - \frac{1}{n(N-n)}\left[\sum_{j=1}^{n} r_j - \sum_{j=1}^{n} j\right]$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$= a\Re_{norm} + b$$

where $a = \frac{N}{N-n}$ and $b = \frac{-(n+1)}{2(N-n)}$

Here the normalized recall function may be expressed as

$$R_{norm} = a\,\Re_{norm} + b$$

$$= \frac{N}{N-n}\left(1 - \frac{1}{Nn}\sum_{j=1}^{n} r_j + \frac{1}{N}\right) - \frac{(n+1)}{2(N-n)}$$

$$= \frac{1}{n(N-n)}\left[Nn - \sum_{j=1}^{n} r_j + n - \frac{n(n+1)}{2}\right]$$

$$= \frac{1}{n(N-n)}\left[Nn - n^2 - \sum_{j=1}^{n} r_j + n + n^2 - \frac{n(n+1)}{2}\right]$$

$$= \frac{1}{n(N-n)}\left[n(N-n) - \sum_{j=1}^{n} r_j + \frac{n(n+1)}{2}\right]$$

$$R_{norm} = 1 - \frac{1}{n(N-n)}\left(\sum_{j=1}^{n} r_j - \sum_{j=1}^{n} j\right) \qquad (2\text{-}50)$$

Equation (2-50) shows the proper dependence on the ranks of the relevant documents and is in fact equivalent to the normalized recall previously derived and shown in eqn (2-44).

So, one finally gets

$$R_{norm} = 1 - \frac{\sum_{i=1}^{n} r_i - \sum_{i=1}^{n} i}{n(N-n)}$$

$$= a\Re_{norm} + b$$

where $a = \frac{N}{N-n}$ and $b = \frac{-(n+1)}{2(N-n)}$

and

$$\Re_{norm} = \frac{1}{N}\sum_{i=1}^{N} R(i)$$

When the number of relevant documents n is small compared to the total number of documents in the collection N, a is close to 1 and b is close to -(1/(2N)), so that

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$R_{norm} \approx \Re_{norm}$

Exactly similarly it can be shown that

$$P_{norm} = 1 - \frac{\sum_{i=1}^{n} \log r_i - \sum_{i=1}^{n} \log i}{\log [N!/((N-n)!\, n!)]}$$

$$= c\, \wp_{norm} + d$$

where $c = \frac{N}{\log \binom{N}{n}}$ $\quad d = 1 - \frac{n \log N + 1}{\log \binom{N}{n}} + \frac{\log (n!)}{\log \binom{N}{n}}$

and $\wp_{norm} = 1 \sum_{i=1}^{N} P(i)$

These parameters c and d are close to 1 and 0, respectively, only for reasonably large N and for n neither very small nor very large. In many situations where n is quite small compared to N, the normalized precision is not an accurate measure of the average standard precision.

2.23 *Methods for determining the recall value*

One of the puzzling things in the above eqn for the recall function is that it depends on the relevant documents retrieved as well as the relevant documents not retrieved (n). So to evaluate the recall it is necessary to assess recall function to each document in the collection irrespective of whether they are retrieved or not. Calculation of the recall value is needed for ranking purpose. But in an operational system, if the size of the collection is very large, assessing recall to each document may be really cumbersome. There are some methods for this.

One commonly used method is *sampling method* which examines a small set of documents to assess the recall value and then applies the results on the whole collection. For example, if 20 documents were found to be relevant in a 10% sample of a collection, then the whole collection has 200 relevant documents. But the problem is that the sampling density may be variable depending on the types of documents and hence the sample may not be a good representative.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Another approach to detect relevant material in the non-retrieved set is to examine subject indexes or keyword-in-context (KWIC) indexes for the collection and to search in these volumes for references to documents that can possibly be relevant.

Here is an example where four search methods have been used for a phrase as "*mantle of the earth*"

as described in table 2-10. In each case, upto five relevant documents are to be retrieved in decreasing order of correlation. These five documents are different for each search and actually a total of seven distinct documents are retrieved as a result of the four searches. If it is assumed that no further relevant documents are present in the collection, the recall will increase from 2/7 after the first search to a value 1 after the fourth iteration. Actually there are totally 10 relevant documents and so the computed recall values are only some approximations of the actual values.

2.24 Presentation of results

Assuming that there are procedures available for calculating both recall and precision values for a search method and a given user query, the following issues may be verified regarding the methods of presenting the assessment results.

1. There are many different analysis, searching and matching procedures available each generating different recall-precision pairs for the same query.
2. A new pair of recall-precision values is generated every time a new cutoff is used to separate the retrieved from the non-retrieved documents. AS a consequence, the retrieval results may no more be comparable between different queries as the cutoff changes.
3. It may be worthwhile to average the recall and precision results over many search queries; so the issue becomes what should be the averaging standards.
4. Sometimes in stead of reporting the numerical figures of recall-precision pairs, it is plotted on a graph for all the search methods; in such case one should get many recall-precision

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

pairs for all the various cutoff value and also the results should be averaged over many queries.

Table 2-10 Approximate recall values using various search methods

Crust of the earth - totally 10 relevant							
Search type	Doc rank	Doc No	Relevant	Cumulative No of distinct relevant	Cumulated precision	Cumulated recall computed	Cumulated recall Correct
Null Thesaurus (Word stem match)	1 2 3 4 5	129 316 167	X X	2	2/5	2/7	2/10
Harris 2 Thesaurus	1 2 3 4 5	316 129 313 176 167	X X X X	4	4/7	4/7	4/10
Statistical phrases	1 2 3 4 5	371 316 372 002 313	X X X X	6	6/8	6/7	6/10
Hierarchy	1 2 3 4 5	264 316 173 218 129	X X X	7	7/9	7/7	7/10

As far as the various search methods are concerned, it may be fair to report the recall-precision pairs for each method separately. As far as the cutoff goes, it is not required in a system where normalized measures on the ranks of the relevant documents are used. Such a choice is, however, required for standard keyword search systems. Several types of cutoff procedures may be suggested:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1. There can be a cutoff of k, such that at least k matching keywords must match between the document and the query for being retrieved.
2. If the documents are ranked, one can decide on a cutoff after retrieving n documents.
3. Alternatively, the cutoff can be applied when the correlation coefficients between documents and queries fall below a certain threshold value or when the first-order differences between the adjacent correlation coefficients exceed a certain value.

Once a single cutoff point has been selected by whatever method, the reporting problem is easy. But when the cutoff point is allowed to vary, or if several search requests are present, a large number of alternative ways of presenting the results become possible.

For example, table 2-11 shows the results after queries X and Y with 10 and 5 relevant documents respectively. Assuming a variable cutoff after 1, 5, 10, 15, 20, 25 documents, one gets for each query six recall-precision points.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 2-11 Recall-precision pairs for two queries

Request X: 10 relevant documents
ranks: 1, 15, 16, 17, 18, 19, 20, 21, 22, 23

Cutoff values	1	5	10	15	20	25
Recall	0.1	0.1	0.1	0.2	0.7	1.0
Precision	1.0	0.2	0.1	0.13	0.35	0.4

Request Y: 5 relevant documents
ranks: 1, 2, 3, 20, 21

Cutoff values	1	5	10	15	20	25
Recall	0.2	0.6	0.6	0.6	0.8	1.0
Precision	1.0	0.6	0.3	0.2	0.2	0.2

At this point, if it is desired to choose a single precision value for each recall point, one can either interpolate for an intermediate cutoff value and use the recall-precision point for that cutoff. As example, for request X and method 1, a cutoff of 7 documents would be used, producing a recall of 0.1 and precision of 0.14.

The averaging operation can be done in two ways. One, the query oriented viewpoint, has been macroevaluation and consists in taking average recall-precision values on a per-query basis (over a range of queries). The average macrorecall and the average macroprecision taken over m queries are given as

$$\text{Average macrorecall} = \frac{1}{m} \sum_{i=1}^{m} \frac{c_i}{b_i + c_i}$$

and

$$\text{Average macroprecision} = \frac{1}{m} \sum_{i=1}^{m} \frac{c_i}{c_i + d_i}$$

where b_i, c_i and d_i are, respectively the number of documents relevant but not retrieved, the number of relevant and retrieved, and the number irrelevant but retrieved corresponding to the I-th query.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The second type of averaging operation is called microevaluation. Here, it is important to be able to distinguish relevant from nonrelevant documents, with the result that search queries with a large number of relevant documents are weighted more heavily than those with few relevant documents. Specifically, given two queries with n_1 and n_2 relevant documents, the microevaluation attempts to locate at each point a total of n_1+n_2 relevant documents. So, the average microrecall and microprecision for m queries are given as

$$\text{Average microrecall} = \frac{\sum_{i=1}^{m} c_i}{\sum_{i=1}^{m} (b_i+c_i)}$$

$$\text{and Average microprecision} = \frac{\sum_{i=1}^{m} c_i}{\sum_{i=1}^{m} (c_i+d_i)}$$

The macro expressions become very different from the micro expressions when no. of relevant documents retrieved are widely different in the m queries, otherwise they produce similar results.

Programmed evaluation In a retrieval situation, it is possible to get recall and precision functions automatically computed as part of the retrieval process. For that reason, it is necessary to introduce for each processed query a list of the corresponding relevant document identifications. Specifically, the requester, or an outside person, is then given a copy of the full document collection after the request is received, and he is asked to list the documents that should be considered relevant to his request. It is important to note that these relevance judgments are a priori judgments, based on the document text only and not on retrieval results produced by the computer. Furthermore, since the same relevance judgments are used to evaluate all the retrieval procedures under investigation, any possible bias introduced by individual faulty relevance judgments may be expected to be in the same direction for all methods. Under such circumstances, results based on the relative difference between the relative performance for two different retrieval methods should remain comparatively unaffected by variations in the relevance judgments. Only the absolute values of the performance measures will vary, with the result that any claim

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

based on the absolute value of either the recall or the precision measures must be phrased with considerable care.

The documents ranks are used in generating many various measures reflecting recall and precision, including the normalized recall and normalized precision measures as explained before. Also calculated are simplified expressions, termed rank recall and log precision and defined as follows:

$$\text{Rank recall} = \frac{\sum_{i=1}^{n} i}{\sum_{i=1}^{n} r_i}$$

$$\text{log precision} = \frac{\sum_{i=1}^{n} \ln i}{\sum_{i=1}^{n} \ln r_i}$$

These measures are like the normalized recall and normalized precision but they do not take into account the collection size N.

Two more composite measures are given here which refer to the recall and precision components. The first one consists of the sum of rank recall and log precision. The other is a weighted sum of the normalized measures, as follows:

Normed overall measure = $1 - 5(R_{norm}) + P_{norm}$

This factor of 5 is so chosen as to give equal weight to the two component measures.

Also it is generally noticed that the recall-precision combination is improved for statistical phrase analysis than ordinary thesaurus lookup procedure.

Output summary To obtain statistically useful measurements, the recall and precision values must be averaged over many different queries. This is being shown in fig 82 for 9 different processing methods and a total of 10 specific and 7 general queries.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

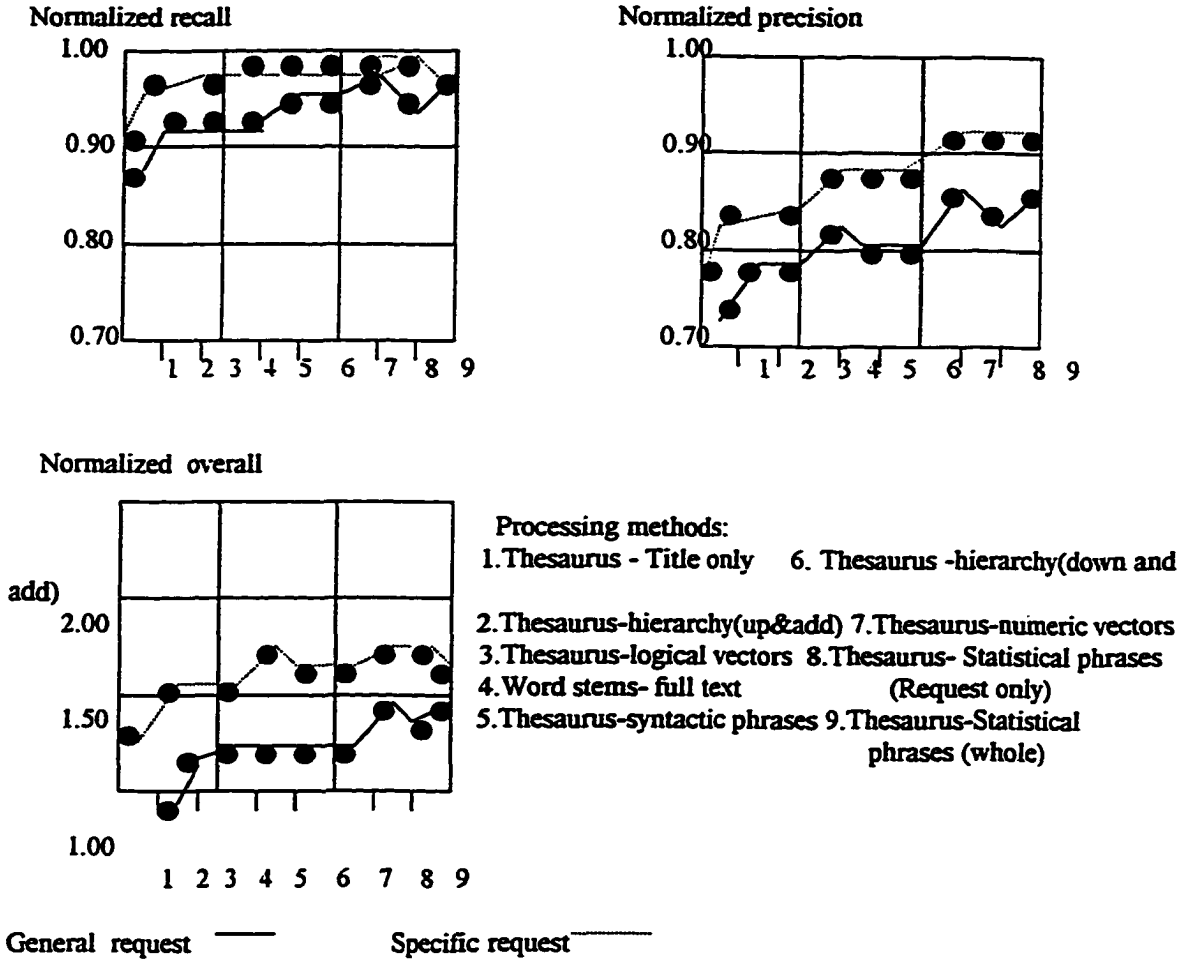


Figure 82:Normalized recall, precision and overall measures

A few observations can be made from the fig 82 above.

1. The normalized functions for the various processing methods show noticeable differences.
2. As one switches from one method to another method, both the normalized recall and normalized precision move in the same direction.
3. All the measures (recall, precision and overall) obtained for very specific type of queries are larger than the corresponding values for the general type of queries, thus proving a better systems performance is always available for the very specific types of queries.
4. Methods 1 through 4 seem to generate relatively poorer recall as well as poorer precision than methods 5 through 9.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5. The use of the regular thesaurus, that offers vocabulary control (method 7), seems more effective, on an average, than the use of the original words included in document and search requests.
6. The more productive algorithms are the ones that use phrases than use individual concepts.

The diagram of fig 82 proves that more effective procedures (rather than only word matching) can be built to improve retrieval performance in an automated retrieval system.

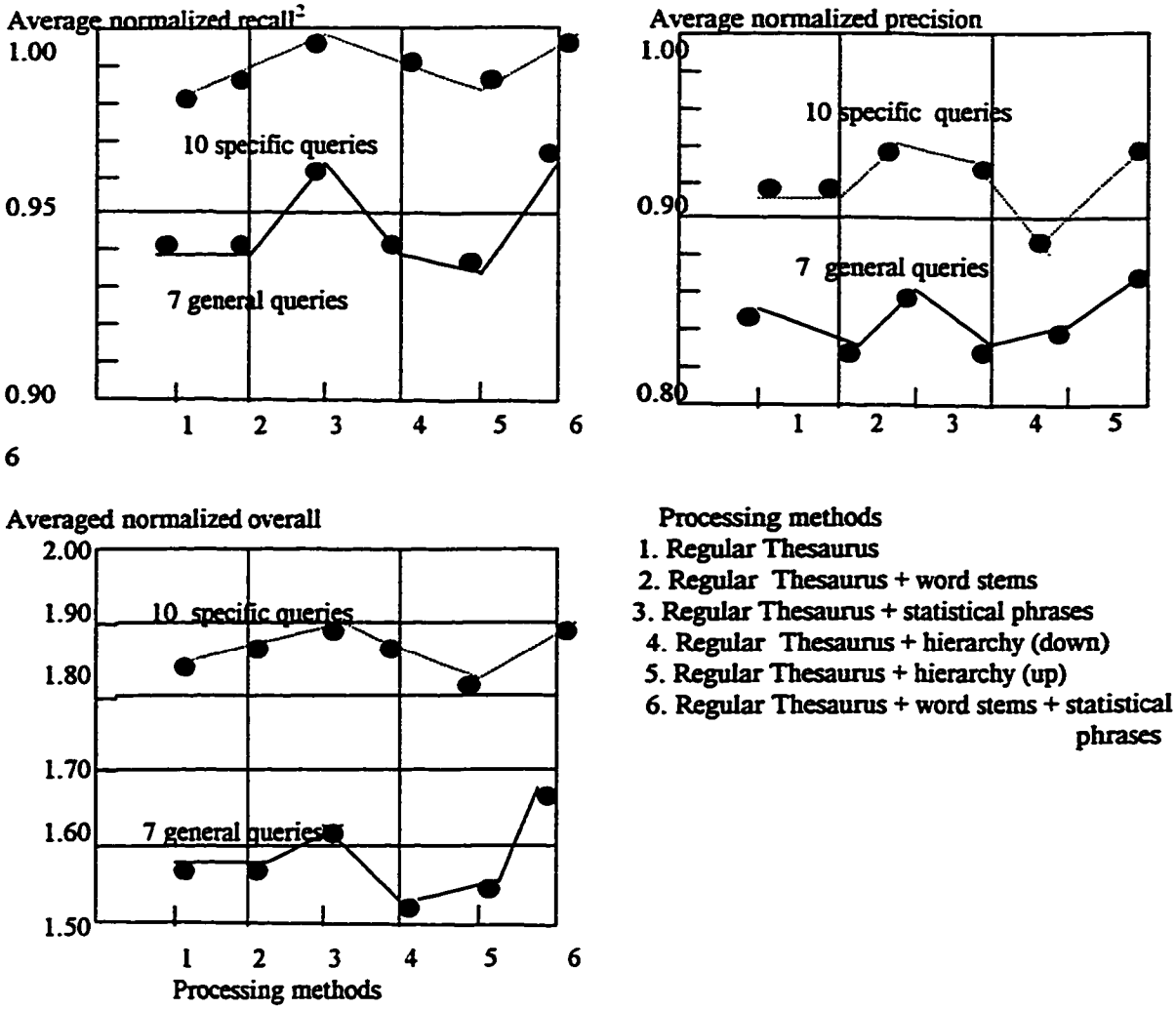


Figure 83:Normalized recall, precision and overall measures for various methods

But, for an even better performance, queries may be processed by several different methods and later merged. To verify such statements, the average normalized recall and precision figures are given in fig 83 above for 6 such combined methods and 17 queries (10 specific and 7 general). Among the 6 combined

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

methods, one method is using the regular thesaurus alone. The others include regular thesaurus as a component. It may be verified that in three cases (methods 2, 3 and 6) the overall measures for both the specific queries as well as the general queries improve compared to the performance of the components. Method 6, consisting of a combination of regular Thesaurus, word stem and statistical phrase analysis seems to give best performance.

The normalized recall and precision measures for the combined methods are calculated by using the rank lists produced by the each individual method alone and then automatically creating a merged rank list for the combination. The combined rank of a given document depends on the ranks of the document using individual methods. Documents are taken alternately off each list (starting from the top) and are added to the combined list. If a document already exists on the new list, it is discarded at the second time. The new list is used to compute the normalized measures. The results are averaged over several queries to generate the graphs in fig 83. Also the standard recall and standard precision functions may be plotted for the purpose of comparison with various systems. The method adopted for this purpose is:

1. A particular standard recall value is chosen (say 0.15).
2. The number of documents that must be retrieved for a given query to achieve the chosen recall value must be determined.
3. Using this value of retrieved documents in step 2, the precision function is calculated for the chosen recall value.
4. The precision values obtained for the retrieved documents are averaged over multiple queries and then a point is plotted in the standard recall-precision graph.
5. The above steps are followed for many different recall values.

---

[2] Macroviewpoint is chosen in the calculations rather than microviewpoint.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Such graph plots are available which compares the different search methods. They in general confirm that the statistical phrase run offers best performance, word-stem comparisons do not compete well with regular Thesaurus runs and "titles only" analysis is generally inferior. It is also noticed that with various methods, the performance (both recall as well as precision) can vary over a range of 30 percent.

Of course there are also many systems that use microviewpoint analysis and so it becomes hard to compare the exact values. But, the qualitative comparison over many methods can still be observed in different systems rather than analyzing only the absolute values.

Significance computations For each query and each processing method, the evaluation procedures that were described before produce 14 possible different statistics, including 4 global statistics (rank recall, log precision, normalized recall and normalized precision) and 10 local statistics (standard precision for 10 recall levels as 0.1, 0.2, 0.3,...., 1.0). The various use of these 14 different statistics certainly creates confusion in the area of system assessment (questions as which one to use and which one not to use etc.). Practically speaking, it is difficult to conclude about methods' performances because the variation among the performance for individual queries is large, the 14 measures have their own ranges and also it is unknown if a certain value for recall or precision is significant or not. In particular, it is of interest to know if values as big as a given value can be expected under random circumstances (i.e. often) or rather oppositely it is expected to happen very rarely.

Due to the wide variance of queries and large differences in the range of various parameters, the significance of absolute value in performance level, becomes unimportant. Rather the paired comparisons are made between query performance for methods X and method Y. The difference in performance level is computed for each of the 14 pairs of statistics available for each query for each pair of processing methods. These differences are then averaged over many queries and then they are transformed into probability measurements. Each of the 14 values thus obtained implies the probability that, if the performance level of the two methods X and Y were equally high for the given statistic(except for some random variations in the test results), then a test value as large as the one observed will occur for the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

system. A probability of 0.05 is usually taken as an upper bound in judging whether a deviation in test values is significant or not. With this probability as a limit, the corresponding test difference would be significant 19 times out of 20, and only one time among 20 times will two equally effective systems be expected to produce such a large test difference.

In addition, since it is difficult to judge systems performance by using 14 different probability values corresponding to the 14 evaluation measures, an aggregate probability value may be computed from the 14 individual probabilities. The significance of this aggregate depends on the independence of the various significance tests.

There are many different testing procedures that exist of which two are principal ones. The first procedure uses the well-known t-test based on student's t distribution. This test requires an underlying normal distribution of the data used in the test process, as well as the independence among the queries processed with document collections. The t-test process takes into account the actual magnitude of the differences for the statistics being calculated, and the resulting probabilities are considered to be reliable indicators of system differences.

A less demanding test procedure is furnished by the sign test, where the magnitude of the differences in the statistics is not taken into account, but only the sign of the differences (i.e., an indication of if method X offers larger test results than method Y or the reverse). In this case the normal distribution is not assumed and as the normal distribution is hard to prove from a query-document correlation process, the sign-test probabilities may offer a better indicator of system performance than the t-test.

The t-test computations are performed as follows: Let m_{ijX} be the value of statistic I for request j, using method X (for example, the value of the rank recall or log precision). Then, given two methods, X and Y and a set of k queries, the average of the differences for statistic I is computed. Specifically,

$$d_{ij} = m_{ijA} - m_{ijB}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and

$$D_i = \frac{1}{k} \sum_{j=1}^{k} d_{ij}$$

The difference computations for two statistics rank recall and log precision are shown in table 2-12. The average differences are then used to obtain the standard deviation (SD) of the differences in the t-test values. In particular,

$$T_i = \frac{D_i}{(SD)_i} \sqrt{k}$$

The t-test values T_i are now converted to probabilities P_{ti}, using Student's t distribution with k degrees of freedom.

The probabilities derived from the 14 statistics can be used to compute an aggregate probability by first converting the two-tailed t test to a one-tailed test, changing each probability to chi-square, adding the chi-square values, and then reconverting to a probability P_t, using a chi-square distribution with 28 degrees of freedom.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 2-12 Calculation of recall and precision differences for individual queries (method X: stem words; method Y: Thesaurus)

Request name	Rank recall			Log precision		
	Method X	Method Y	Difference	Method X	Method Y	Difference
Automata Phr	0.5238	0.9649	-0.4411	0.7126	0.9881	-0.2755
Comp Systems	0.0725	0.1228	-0.0503	0.3783	0.4806	-0.1023
Comps-Assemb	0.3714	0.7428	-0.3714	0.8542	0.9453	-0.0911
Core memory	0.0691	0.1064	-0.0373	0.3157	0.3695	-0.0538
Differentl Eq	0.5298	0.7574	-0.2276	0.8620	0.9219	-0.0599
Error control	0.1460	0.1875	-0.0415	0.5342	0.5972	-0.0630
M10-counters	0.8182	0.7347	0.0835	0.8682	0.8599	0.0083
M2-transmit	0.0522	0.0963	-0.0441	0.2819	0.4698	-0.1879
M3-inform	0.1968	0.3134	-0.1166	0.6300	0.7666	-0.1366
M8-storage	0.0375	0.2763	-0.2388	0.2670	0.4666	-0.1996
Missile trak	1.0000	0.7500	0.2500	1.0000	0.6309	0.3691
Morse code	1.0000	1.0000	0.0000	1.0000	1.0000	0.0000
Pattern recg	1.0000	1.0000	0.0000	1.0000	1.0000	0.0000
Random numbs	0.0517	0.2000	-0.1483	0.1750	0.3408	-0.1658
Solstat circ	0.2766	0.3402	-0.0636	0.6921	0.7912	-0.0991
Switch funcs	0.3529	0.4444	-0.0915	0.7416	0.8005	-0.0589
Thin films	0.2157	0.8462	-0.6305	0.6294	0.9242	-0.2948
Total	6.7142	8.8833	-2.1691	10.9422	12.3531	-1.4109
Average value over 17 queries	0.3950	0.5225	-0.1276	0.6437	0.7267	-0.0830

The t-test computations are shown for two sample analysis methods X and Y in table 2-13. The values in the first two columns of table 2-13 represent averages over 17 queries for each of the 14 assessment functions. The final probabilities P_{ti} range from a high of 0.107 for the standard precision at recall value 1, to a low of 0.0007 for the normalized precision. The final probability value P_t is smaller than 1×10^{-4}, thus indicating that the combination algorithm concentrates on the significant tests, while ignoring the less significant ones. The validity of the process depends on an assumption of independence of the 14 functions which is only partly true.

The sign-test uses the binomial instead of the t distribution to produce a probability value. Specifically, given two processing methods for which the null hypothesis applies (i.e. two methods are

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

equivalent), each d_{ij} has equal chances of being positive or negative; moreover since the queries are assumed unrelated (independent) and randomly distributed, the signs of the differences are unrelated. The number, say M, of positive signs is accordingly binomially distributed with p equal to ½ and k equal to the number of requests. Then M can be used as an important statistic to test the null hypothesis by taking large values of M as the evidence contradicting the assumption of equivalence of the methods.

Of course, a test based on rejecting the equivalent hypothesis for large values of M is equivalent to rejecting for small values of M', the number of negative signs. As before, a probability of 0.05 may be chosen as an upper limit for rejecting equivalence assumption.

Since the sign-test does not depend on the magnitudes of the differences, the number of positive or negative signs can be accumulated directly. In particular, the number of queries preferring method X is summed over all measures as well as the number of queries preferring method Y. These totals are then subjected to the same testing process, as follows: Let t be a tolerance value, taken as 0.001 for the current test; further for each statistic I, let

k_{ai} = number of d_{ij} (j=1, ...,k) exceeding +t

k_{bi} = number of d_{ij} smaller than -t

k_{ci} = number of d_{ij} such that $|d_{ij}| \leq t$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 2-13 t-test calculations for 14 different recall and precision measures
(Averages over 17 queries: method X, stem word; method Y, Thesaurus)

Evaluation measure		Average value Method X	Average value Method Y	Difference of average	Standard deviation	t-test value	Probability
Rank recall		0.3950	0.5225	-0.1276	2.07 E -01	2.54 E -00	0.0219
Log precision		0.6437	0.7267	-0.0830	1.47E -01	2.33 E -00	0.0334
Normalized recall		0.9233	0.9675	-0.0442	5.35E -02	3.41 E -00	0.0036
Normalized precision		0.7419	0.8639	-0.1219	1.20E -01	4.19 E -00	0.0007
Local Recall-precision points (Precision at 10 different levels)	0.1	0.7385	0.9735	-0.2351	2.88 E -01	3.37 E -00	0.0039
	0.2	0.6544	0.8973	-0.2428	2.82 E -01	3.55 E -00	0.0026
	0.3	0.5844	0.8245	-0.2401	2.51 E -01	3.95 E -00	0.0011
	0.4	0.5326	0.7551	-0.2226	2.39 E -01	3.84 E -00	0.0014
	0.5	0.5187	0.7146	-0.1959	2.00 E -01	4.04 E -00	0.0009
	0.6	0.5035	0.6499	-0.1464	1.59 E -01	3.79E -00	0.0016
	0.7	0.4452	0.6012	-0.1561	1.79 E -01	3.59E -00	0.0024
	0.8	0.4091	0.5514	-0.1423	2.24 E -01	2.62E -00	0.0184
	0.9	0.3794	0.4973	-0.1179	2.29 E -01	2.12E -00	0.0499
	1.0	0.3106	0.4118	-0.1012	2.44 E -01	1.71E -00	0.1070
Combined significance: total chi-square = 1.67 E -02 Total probability of Y over X = 0.0000							

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 2-14 Sign-test for 14 different recall and precision measures
(Averages over 17 queries: method X, stem word; method Y, Thesaurus)

Assessment measure		Number of requests superior for method X	Number of queries superior for method Y	Number of queries equal for methods X, Y	Probability of of Y over X
Rank recall		2	13	2	0.0074
Log precision		2	13	2	0.0074
Normalized recall		2	13	2	0.0074
Normalized precision		2	13	2	0.0074
Local Recall-precision	0.1	0	9	8	0.0039
	0.2	0	11	6	0.0010
	0.3	1	12	4	0.0034
	0.4	1	11	5	0.0036
	0.5	0	13	4	0.0002
	0.6	3	11	3	0.0574
	0.7	2	12	3	0.0129
	0.8	3	12	2	0.0352
	0.9	4	11	2	0.1185
	1.0	4	11	2	0.1185
Combined significance for 14 measures		26	165	47	0.0000

where the number of requests $k = k_{ai} + k_{bi} + k_{ci}$

The sign-test probability of statistic i is now computed as follows:

Let $k_{vi} = k_{ai} + k_{bi}$

and

$$k_{wi} = \min (k_{ai}, k_{bi})$$

Then

$$P_{si} = \sum_{j=0}^{k_{wi}} \frac{k_{vi}! \quad 2^{-k_{vi}+1}}{j! \, (k_{vi} - j)!}$$

The sign-test calculations are shown in table 2-14 above for the same processing methods and queries that were used in table 2-12 and table 2-13. The individual probabilities P_{si} range in values from 0.0010 to

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

0.1185. The overall probability is again smaller than 1×10^{-4} ; this is evident from the fact that Y is superior over X 165 times where X is superior only 26 times with 47 ties.

Since all the 14 statistics may not be fully independent, a question comes up on the interpretation of the accumulated t-test probability P_t and the accumulated sign-test probability P_s. As a general rule, the equality hypothesis between two given methods X and Y can be rejected when both probabilities P_s and P_t do not exceed 0.001 in value, implying that most of the individual probabilities P_{si} and P_{ti} are smaller than 0.05, and when the same test results are obtained for all documents being tested. If, on the other hand, the values of the final probabilities are larger, or if the test results differ from one collection to the next, additional test may be needed to reach a decision.

2.25 Analysis of Automated information systems

The principles of assessment as described above can be used as a basis for the analysis of the various features of automated information systems. Such an objective analysis leads one to the design of a more effective system by answering many questions as 1) can automated text analysis method replace manual content analysis and indexing; 2) is it necessary to offer vocabulary normalization methods to eliminate ambiguities caused by homographs; 3) is it worthwhile to have word-stemming as opposed to a synonymous dictionary; should such normalization be handled by a pre-constructed dictionary or is it possible to replace thesauruses completely by statistical word association methods; 4) what type of dictionaries will be most helpful for automated information retrieval? 5) Are the hierarchical processing methods any good etc. Many such relative questions can be answered better using the comparative mathematical analysis of the various statistics. It is always better to implement a procedure that favors retrieval of relevant materials as well as precise materials. The results of various kinds of processing have been presented here on basis of three different document collections in three different subject areas.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.25.1 Test environment

The major parameters controlling the test procedure are listed in tables 2-15 to 2-17. The main properties of the document collections and search queries are shown in table 2-15. Specifically, results are given for document collections in three different subject areas;

1. Computer Science (IRE-3): A set of 780 abstracts of documents in the computer literature, used with 34 queries.
2. Documentation (ADI): A set of 82 short papers, each an average of 1380 words in length, presented at the 1963 Annual meeting of the American Documentation Institute, and used with 35 queries.
3. Aerodynamics (Cranfield-1): A set of 200 abstracts of documents previously searched by the second Aslib-Cranfield project and used with 42 queries.

Each of these collections belongs to a separate subject area, thus permitting the comparison of the

various analysis and search methods in different texts. The ADI collection in documentation is of particular interest because the full texts are available in stead of abstracts only. The Cranfield collection is also manually indexed by trained indexers, thus rendering it possible to compare between manual indexing and automated analysis. The queries are in short paragraphs in English. For two collections, the queries were created for test purpose only, whereas the Cranfield queries are from actual users in the field of Aerodynamics. The procedure used for collecting relevance assessments and the related statistical information regarding the average number of relevant documents per request are summarized in table 2-16.

Detailed procedures were used to assess the relevance of each document with respect to each search query. Only one person was asked to collect the judgments of each request, and binary assessments (relevant or not) were made for each document. The dichotomous (as opposed to grades of relevance) judgments were delivered by a group of people privately and the average of those private judgments have

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

been used as the indicator of relevance for each document. Additional data regarding the user program and the number of search programs used are given in table 2-17.

Table 2-15 Document collection and query characteristics

Characteristics			IRE-3	Cranfield-1	ADI
Document collection	Number of documents in collection		780	200	82
	Average number of words (All words) per document	Full-text	...	...	1380
		Abstract	88	165	59
		Title	9	14	10
	Average number of words (common words deleted) per document	Full-text	...	...	710
		Abstract	49	91	35
		Title	5	11	7
	Average number of concepts per analyzed document	Full-text	...	...	369
		Abstract	40	65	25
		Title	5	9	6
Search queries	Number of search queries		34	42	35
	Average number of words per query (all words)		22	17	14
	Query preparation: a. Short paragraphs prepared by staff members for testing only b. Short paragraphs prepared by subject experts As part of regular operations.				

In each case, The user population was made of about 10 students cum subject experts and about 42 professional subject experts, all as volunteers. About 60 different analysis and search methods were used in various tests, results of which are presented here. The following major parameters have been investigated: length of the document being analyzed; matching algorithms for the comparison of documents and query processing (including the use of weighted terms and a variety of correlation coefficients); language normalization methods; synonym dictionaries, phrase dictionaries and statistical term association methods based on the principle of co-occurrences of terms in the various documents; hierarchical subject classifications; keyword indexing based on manual choices of terms; and the use of iterative searches where increasingly more refined queries are built based on feedback information.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Document Length:

The length of document to be used for content analysis is a variable of interest. The issue of whether to use full content or not is a controversial topic because many believe that manual content analysis is essential as total content analysis is not possible on an automated level. But instead, the analysis may be restricted to the titles only or keywords-in-context (KWIC), (i.e. permuted titles). In the test, three analysis systems based on document titles were compared with systems based on processing full abstract. In each case, the weighted word stems, extracted from the titles or from the abstracts are compared with the equivalent indicators from the queries.

Table 2-16: Relevance distribution and assessment

Characteristics	IRE-3	Cranfield-1	ADI
Preparation of relevance judgments:			
a. Dichotomous, prepared by staff experts based on abstracts, using full relevance assessment.	√		
b. Dichotomous, prepared by subject experts based on abstracts and full-text (full relevance assessment)		√	
c. Dichotomous, prepared by staff experts based on full-text, using full relevance assessment.			√
Number of relevant documents per request (all requests):			
a. Range	2-65	1-12	1-33
b. Mean	17.4	4.7	4.9
c. Generality (mean divided by collection size)	22.2	23.6	59.2
Number of relevant documents per specific query:			
a. Number of specific requests	17	21	17
b. Mean number of relevant	7.5	3.0	2.1
Number of relevant documents per general query:			
a. Number of general queries	17	21	18
b. Mean number of relevant	25.8	6.4	7.4

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 2-17 General test environment

Characteristics	IRE-3	Cranfield-1	ADI
User population:			
a. 10 students and staff experts	√		√
b. 42 subject experts		√	
Number of retrieved documents per query	All	All	All
Number of indexing and search programs used:			
a. Matching algorithms	2	2	2
b. Term weight adjustment	2	2	2
c. Document length variation	3	4	3
d. Basic dictionaries (suffix s, stem, thesaurus, statistical phrases, hierarchy, syntax)	6	5	5
e. Concept-concept association dictionaries	2	3	1
f. Total basic options	144	240	60

Table 2-18 Significance output for title-abstract comparison

Assessment parameters	IRE-3	Cranfield-1	ADI
Number of tests			
(A) favoring title	89	195	156
(B) favoring abstract	330 √	333 √	276 √
favoring neither	57	60	58
Probability P_s (B/A) (B over A)	0.0000	0.0000	0.0000
Probability P_t (B/A) (B over A)	0.0000	0.0000	0.0000

Tests were conducted to measure recall-precision averaged over many queries in computer science, aerodynamics and documentation collection respectively (titles vs abstract). In all the cases, the abstract process showed better recall-precision combination than the "title-only" option. At the low recall-area, the precision appears to be the same, but as normalized recall goes above 0.4. the gap widens for the precision increasing further until recall value of 1. The superiority of abstract-stem process (using cosine correlation and numeric vectors) is evident in table 2-18. The probability of a correct null hypothesis (that A and B are equivalent) is smaller than 1. 10^{-4} for both the sign and t-test and it also shows that document titles are definitely inferior to abstracts treated for content identifiers.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The ADI documentation collection was used to even extend the analysis further to full text. There was test done with abstract processing (only 60 words) as opposed to full-text processing (1400 words) using word-stems in one case and a synonym Thesaurus in the other case. In the synonym thesaurus, all the synonymous word-stems are looked up before the document identifiers are matched with the query identifiers. The result is that the full text gives better recall-precision than the abstract process although the improvement is not very prominent.

Matching Functions and term weights

In an automatic text processing environment it is easy to differentiate among individual content indicators by assigning weights to the indicators in proportion to their importance. Such weights can be derived in part by using the frequency of occurrence of the original words and in part as a function of various dictionary mapping procedures.

Table 2-19 Significance output for numeric vs. Logical weights

Assessment parameters	IRE-3	Cranfield-1	ADI
Number of tests			
(A) favoring numeric	295 √	308√	238√
(B) favoring logical	129	149	218
favoring neither	52	131	34
Probability P_s (A/B)	0.0000	0.0000	0.3736
Probability P_t (A/B)	0.0000	0.0000	0.0040

In this way, ambiguous terms that will appear in many groups in the synonym dictionary can be weighted less than the unambiguous terms. Performance of such weighted (numeric) content-identifiers were compared with the logical-identifiers (where all terms have values 1 or 0). The significance output for the test is presented in table 2-19. The numbers are clearly favoring weighted terms although in the case of documentation collection, as many assessment parameters favor the non-weighted process as the weighted process. But as a whole, the weighted content indicators show better retrieval results than the non-weighted identifiers.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Correlation coefficient

One important variable in information retrieval is the choice of the correlation coefficient. Two important functions are the cosine function and the overlap function.

$$\text{Cos}(q, d) = \frac{\sum_{i=1}^{n} d_i \cdot q_i}{\sqrt{\sum_{i=1}^{n} (d_i)^2 \cdot \sum_{i=1}^{n} (q_i)^2}}$$

and

$$\text{overlap}(q, d) = \frac{\sum_{i=1}^{n} \min(q_i, d_i)}{\min\left(\sum_{i=1}^{n} q_i, \sum_{i=1}^{n} d_i\right)}$$

where q and d are the n-dimensional vectors of terms for the query and an analyzed document respectively.

Both of these functions range from 0 for no match to 1 for exact equality of the vectors. The cosine is very sensitive to the vector's length (due to the denominator) and tends to produce greater variation than the overlap function. A list of the significance output for some test is given in the table 2-20. In each case the binary vectors are used with either of the two correlation functions. The results are in favor of the cosine functions although the sign-test result for the ADI collection is not sufficiently one-sided to reach a hard conclusion.

Table 2-20: Significance output for cosine vs. Overlap correlation

Assessment parameters	IRE-3	Cranfield-1	ADI
Number of tests			
(A) favoring cosine	267√	330√	230√
(B) favoring overlap	165	148	194
favoring neither	44	110	66
Probability P_S (A/B)	0.0000	0.0000	0.0891
Probability P_t (A/B)	0.0000	0.0000	0.0148

Table 2-21 shows significance output for the correlation and weight combination. The weakest combination is the overlap function with logical vectors. And most satisfactory result comes from numeric

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

vectors (weighted) using cosine correlation function. The weighted overlap function is equivalent to the logical overlap function. The equivalence probabilities are less than 1×10^{-4} in the last two columns and the evaluation is fully significant. Thus weighted content-identifiers are more effective than binary weights and the cosine correlation is more effective than overlap correlation function.

Table 2-21 Significance output for correlation and weight combination

Cranfield-1, abstracts, word-stem match

Assessment parameters	(A) Overlap-Num (B) Overlap-Bin	(A) Overlap-Bin (B) Cosine- Bin	(A)Cosine-Bin (B)Cosine-Num	(A)Ovrlp-Num (B)Cosine- Num
Number of tests				
favoring A	24	148	149	130
favoring B	32√	330√	308√	350√
favoring neither	532	110	131	108
Probability P_s (B/A)	0.3497	0.0000	0.0000	0.0000
Probability P_t (B/A)	0.1427	0.0000	0.0000	0.0000

Language normalization

Language normalization remains the primary concern of document analysis when natural language texts are being processed using the automated analysis procedures. Many various dictionaries are in use for language normalization as following.

1. Suffix s process, where words differing by the addition of an s at the end are considered equivalent; for example, orange and oranges are equivalent.
2. The word stem dictionary, where all words pertaining to the common word-stem are considered equivalent. For example, synthesis, synthesize, synthesizer are equivalent.
3. The synonym dictionary, or Thesaurus, where a set of closely related terms are grouped into a class, thus making sure any of the terms is recognized as equivalent. This increases the scope of matching.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4. The statistical phrase dictionary, which makes it possible to recognize "phrases" that consist of juxtaposition of many concepts. Thus if a document contains the concepts as communication as well as systems, it could be tagged as a phrase 'communication systems';
5. The concept association method, in which the concepts are grouped not using a pre-constructed dictionary, but by using a statistical co-occurrence characteristics of the vocabulary.

In processing a suffix s dictionary, the final s endings are dropped from nouns. But for a word-stem dictionary, there is a complete suffix dictionary and a set of rules.

Table 2-22: *Significance output for word-stem vs. Suffix s comparison*

Assessment parameters	IRE-3	Cranfield-1	ADI
Number of tests			
(A) favoring word stem	272 √	134	254√
(B) favoring suffix s	132	371√	107
favoring neither	72	183	129
Probability P_s	0.0000(A/B)	0.0000(B/A)	0.0000(A/B)
Probability P_t	0.0000(A/B)	0.0000(B/A)	0.0000(A/B)

And the original word is reduced to the word-stem. The table 2-22 shows the significance output for a comparison between word-stem and suffix s process. The discrepancy between the IRE and ADI results, on one hand and the Cranfield results on the other hand may happen due to differences in vocabularies. Surely, the Cranfield texts are more technical and also the collection is more homogenous than the other collections. To be in a position to distinguish between document abstracts, it is necessary to remember the fine distinctions between the words in Cranfield case (than for ADI or IRE) and these finer differences are lost when multiple words are placed into the same group by going through the suffix cutoff process. Hence it may be concluded that the dictionaries and word-stemming is effective if the vocabulary is redundant and relatively non-technical; otherwise it may not offer any greater benefits.

Synonym recognition

To treat the effect of language variability meaning the same thing, synonym dictionaries have been used. Many experiments have been performed using a variety of synonym dictionaries for each of the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

three subject fields under study ("Harris 2" and "Harris 3" dictionaries for computer science, quasi-synonym or QS, lists for Aeronautics Engineering; and regular Thesaurus for documentation). Use of such a synonym dictionary allows replacement of various related terms by similar concept identifiers, thus retrieving the document that deals with "manufacture of switching devices" when the query was "production of solid-state gates". Test that compares the word-stem matching procedure with a method that includes a thesaurus look-up procedure for the recognition of synonyms, shows that suitably constructed synonym dictionaries provide great performance gain. The synonym recognition seems to benefit the Cranfield-1 area also (though marginally).

Table 2-23: Significance output for the Thesaurus-word stem comparison

Assessment parameters	IRE-3	Cranfield-1	ADI
Number of tests			
(A) favoring Thesaurus	312 √	277√	277√
(B) favoring word stem	110	208	140
favoring neither	54	103	73
Probability P_s (A/B)	0.0000	0.0020	0.0000
Probability P_t (A/B)	0.0000	0.1483	0.0000

The table 2-23 shows the significance output for the comparison between synonym thesaurus and word stem processing. This shows that all thesaurus improvements are fully significant, with the exception of the t-test for the Cranfield collection. In case of Cranfield the null hypothesis can not be rejected. So it appears that not all synonym dictionaries lead to a gain in performance. For it to be useful, the dictionary must be built using some solid construction principles. These principles can be embodied into programs using word-frequency lists and concordances derived from sample document collection. The differences in search effectiveness for two sets of synonyms dictionaries are presented through the significance output as in table 2-24. The less effective dictionaries are Harris 2 for IRE collection and Old Quasi-Synonym for the Cranfield. The other two dictionaries (Harris 3 in case of IRE-3 and new Quasi-Synonym in case of Cranfield-1) are the improved versions using the dictionary construction principles. Table 2-24 shows significant improvements from one dictionary version to the other. The conclusion is better dictionaries with synonym recognition provide statistically significant improvements in performance compared to

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

word-stem matching procedure. The improvement is best for the dictionaries that follow the important principles of using word frequency list and concordances derived from sample document collection regarding grouping of the words.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 2-24: Significance output for dictionary comparison
Dictionaries: Harris 2, Harris 3, Old Quasi-Synonym Thesaurus,
New Quasi-Synonym Thesaurus

Assessment parameters	IRE-3 (A) Harris 2 (B) Harris 3	Cranfield-1 (A) Old QS (B) New QS
Number of tests		
favoring A	136	130
favoring B	277	334
favoring neither	63	124
Probability P_s (B/A)	0.0000	0.0000
Probability P_t (B/A)	0.0000	0.0000

Phrase recognition

Recognition of phrases (a collection of words) can be used to identify documents and search requests, rather than only individual concepts alone. Phrase recognition can work using many principles: for statistical phrase recognition, a phrase can be assigned when the specific components co-occur in a given document or in a given sentence (irrespective of their sequence); alternatively, more restrictive phrase generation method can be added by including a syntactic recognition procedure to check certain syntactic compatibility among the phrase components before a phrase is actually accepted.

A statistical phrase dictionary is commonly used at many operations for treating the phrases specially. Also the weights assigned to a located phrase is important to consider. To attach more importance to phrases, the weight assigned to a phrase may be 1.5 times that of single concepts. Tests have been conducted for phrase lookup procedure (using a Harris 3 phrase dictionary) compared with the equivalent procedure for synonym dictionary (using a Harris 3). The significance output of statistical phrase comparison in table 2-25 shows that the improvements are not significant and also it has been found that the phrase dictionary does not provide any real help in the middle recall range. This may be due to recognition of false phrases where the phrase components exist but do not belong to a phrase. But the main deficiency seems to be the limited size of the statistical phrase dictionary. Also phrases in a document can be discovered not through using a dictionary, but exploiting the statistical properties of the words in a text. If two terms co-occur in many sentences or in the documents, then a non-zero correlation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

coefficient can be computed as a function of the number of co-occurrences. If the co-efficient is high, the two terms can be grouped as a phrase, and can be assigned jointly to documents and queries.

Table 2-25: Significance output for statistical phrase comparison

Assessment parameters	IRE-3	Cranfield-1	ADI
Number of tests			
(A) favoring thesaurus	21√	256√	134
(B) favoring statistical phrase	20	174	171√
favoring neither	435	158	185
Probability P_s	1.0000(A/B)	0.0001(A/B)	0.0391(B/A)
Probability P_t	0.8645(B/A)	0.1120 (A/B)	0.1171 (B/A)

Associative methods are highly comparable to thesaurus procedures in performance with an exception. The word associations reflect vocabulary of the collection, whereas the thesaurus can be used in generality and is portable from collection to collection.

There are many various ways of generating the statistical word associations. Specifically, by adjusting many parameters, various types of term associations can be created. Also once an association between term pairs is created, its weight can be varied to lessen or amplify its effect. Two main parameters that can be used in this regard are the cutoff value K in the association coefficient, below which an association among terms is not recognized and the frequency of occurrence of the terms being correlated. When all terms are correlated, no matter how low the frequency of occurrence is, lots of spurious associations also may also be found. If the cutoff for frequency of occurrence is high, some real association may drop out. Increasingly more restrictive association procedures, applied first to words in the frequency range 3 to 50 and later in the frequency range 6 to 100, eliminate many spurious associations.

Tests have been done to compare word stem matching with statistical term-term association method (called stem concon meaning a concept-concept association in which word stems are manipulated). Table 2-26 indicates the significance output for such comparison with frequency range as 6 to 100 and K=0.45.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 2-26: Significance output for word stem vs. stem concon comparison

Assessment parameters	IRE-3	Cranfield-1	ADI
Number of tests			
(A) favoring word stem	199√	231	283
(B) favoring stem concon	173	292 √	291√
favoring neither	104	65	14
probability P_S	0.1948(A/B)	0.0086(B/A)	0.4420(B/A)
probability P_t	0.0566(A/B)	0.0856(B/A)	0.6521(B/A)

This table indicate that, for the Cranfield collections in particular, provide some improvement over the word stem method; also ADI and IRE collection display some improvement over local ranges. Only for Cranfield the sign test shows some significance, otherwise in general no strong claims can be made for the overall effectiveness in general. Also a test was conducted to compare a synonym thesaurus with concept-concept association method. The powerful thesaurus was superior for both the ADI and Ire collections.

Table 2-27: Significance output for thesaurus vs. Stem concon comparison

Assessment parameters	IRE-3	Cranfield-1	ADI
Number of tests			
(A) favoring thesaurus	315√	231	277√
(B) favoring stem concon	111	286√	173
favoring neither	50	71	40
Probability P_S	0.0000(A/B)	0.0175(B/A)	0.0000(A/B)
Probability P_t	0.0000(A/B)	0.6906(B/A)	0.0003(A/B)

For Cranfield, the advantage lies with word stem association process specially at the high recall zone where the more exhaustive indexing procedure represented by the stem associations supplies additional information identifiers and helps to carry on the finer distinctions among the Cranfield documents. The table 2-27 shows exactly the same point. The output suggests that the more restrictive association processes are more effective than more general ones. Specifically, as the number of generated association pairs grows, too many of them become spurious thus eating away the retrieval performance. The sign-test output of table 2-28 shows that the differences in performance between the various association methods

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

seem of some significance, so that in practice, limitations should be imposed on the number and types of associated terms actually used. The conclusion is the phrase generation methods, whether implemented by dictionary look-up or by statistical association processes, offers improvement in retrieval effectiveness for some recall levels; the improvement is not sufficiently general or substantial when averages over many search queries are considered. These may be used in special situations where ad hoc control procedures are helpful.

Table 2-28: Significance output for stem concon parameter variations

Assessment parameters	Cranfield-1, abstracts, stem concon	
	(A) Freq. All, K =0.60 (B) Freq. 3-50, K=0.60,	(A) Freq. 3-50, K=0.60 (B) Freq. 6-100, K=0.45
Number of tests		
Favoring A	197	195
Favoring B	291	280
Favoring neither	100	113
Probability P_s	0.0000 (B/A)	0.0001 (B/A)
Probability P_t	0.0525 (B/A)	0.0991 (B/A)

Hierarchical expansion

Hierarchical arrangements of keyword identifiers are also included in many non-conventional information systems. Subject hierarchies are useful for the representation of generic inclusion relations between terms, and they serve to broaden or narrow down or otherwise "expand" a given description (by including twin concepts). Specifically, given an entry point in the hierarchy, it is possible to find more general terms by going "up" in the hierarchy (expanding by parents) and more specific ones by going "down" (expanding by all sons); related terms with the same parent can be used (expanding by twins) and finally any cross-references between individual entries can be identified.

A hierarchical arrangement of thesaurus entries was used in a test with the IRE collection to verify the impact of the hierarchical expansion procedures. In the test procedure, the concepts present in the document vector and query vector were looked up in the hierarchy and then the properly related (say by 1 step up , or by 2 steps down etc.) hierarchy entries were added to the content identifiers. The

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A hierarchical arrangement of thesaurus entries was used in a test with the IRE collection to verify the impact of the hierarchical expansion procedures. In the test procedure, the concepts present in the document vector and query vector were looked up in the hierarchy and then the properly related (say by 1 step up , or by 2 steps down etc.) hierarchy entries were added to the content identifiers. The expanded document vectors were matched with the expanded query vectors and the relevant documents were arranged in decreasing order of correlation value. A comparison in performance (recall-precision) was studied for a standard thesaurus process and the four types of hierarchy expansions and the corresponding significance output is presented in the following table 2-29.

It is seen that in each case the standard thesaurus is undoubtedly superior; moreover the equality hypothesis can be discarded undoubtedly for the expansions by twins, sons, cross-references. A question remains for expansion by parents, where more general terms are added to the original identifiers. It has been seen that this particular process does in fact improve the retrieval performance for certain recall levels.

Table 2-29: Significance output for hierarchy comparisons with standard thesaurus

Assessment parameters	IRE-3, abstracts, thesaurus			
	(A) Thesaurus (B) Hier. Parent	(A) Thesaurus (B) Hier. Twins	(A) Thesaurus (B) Hier. Sons	(A) Thesaurus (B) Hier.cross-ref
Number of tests				
Favoring A	224√	331√	309√	253√
Favoring B	190	100	109	168
Favoring neither	62	45	58	55
Probability P_S	0.1047 (A/B)	0.0000 (A/B)	0.0000 (A/B)	0.0000 (A/B)
Probability P_t	0.1676 (A/B)	0.0000 (A/B)	0.0000 (A/B)	0.0000 (A/B)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

There are many alternative ways to use a hierarchy. It is possible to expand only the query vectors without expanding the document vectors or vice versa. Terms gotten from the hierarchy can be used to replace in stead of adding to the vectors. One can go up or down by one step or many steps and use all the terms thus derived to modify the queries; In general, these expansions tend to produce large-scale disturbance on the original vector. Sometimes they may be helpful when the query is not directed well (the addition of the hierarchical parents tend to smoothen the direction) specially when the topic of the query is far from the major subjects covered in the collection. It certainly holds some promise for improvements, but then the resolution of the hierarchy must be so high so that the parent and the child are highly correlated. If the correlation coefficient between the parent and the child is not so high, then it rather creates disturbances.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

So it is hard to say that automated text processing is really much below manual indexing in performance; it becomes more meaningful when one takes additional variability in manual indexing into account by considering the subjective nature of such indexing and inconsistent result expected.

Table 2-30: significance output for manual indexing comparison

Assessment parameters	Cranfield-1, manual indexing		
	(A) Index stem (B) Abstract stem	(A) Index stem (B) stem concon	(A) Index thesaurus (B) Abstract thesaurus
Number of tests			
Favoring A	267√	292√	283√
Favoring B	222	221	213
Favoring neither	99	75	92
Probability P_S	0.0465 (A/B)	0.0020 (A/B)	0.0019 (A/B)
Probability P_t	0.0415 (A/B)	0.0176 (A/B)	0.0001 (A/B)

Iterative searching

At present most systems accept a query at a time, processes the query, replies the results and takes the next query. The second query is considered totally disjoint from the previous in content. This

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

model may not be ideal because it assumes that a single information analysis and a single search will suffice for a query. Also a query can not be treated in the same way because the users' needs are subjective where some users want exhaustive treatise and some want only some reference in the document. Thus it is felt that a user should be given some individualized services attuned to his need.

Accordingly, some iterative method may be implemented for replying to the user query, where a user first obtains some output in response to his query, and depending on which documents he browses through, the system learns the types and presents more documents of the similar type. The most effective procedure tried so far is the relevance feedback process where the user returns to the system a list of document numbers previously retrieved, along with the information concerning the usefulness of each document for his search purpose. The system now adjusts the original query by increasing the weight of query terms originally contained in documents identified as relevant and decreasing the weight of query terms contained in the non-relevant set. This process may be repeated a few times each time resulting in modifying the query towards the relevant document set away from non-relevant set.

Search results for the iterative feedback methods, show that large-scale improvements in recall-precision combination is obtained for the first step of query modification and the second iteration offers further increase in effectiveness.

Conclusion on assessment

A summary of the main evaluation is presented in table 2-31 below for eight different retrieval methodologies with respect to the three collections used. The measure used to rank the output is a combined coefficient as the sum of average normalized recall and average normalized precision. The following remarks can be made on the basis of the table 2-31.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 2-31: Overall evaluation for eight processing methods

Order	IRE-3			Cranfield-1			ADI		
	Dictionaries Used	Method	Merit	Dictionaries Used	Method	Merit	Dictionaries Used	Method	Merit
1	D4	Stat. Phrase	1.686	D3	Thesaurus	1.579	D4	Stat. Phrase	1.456
2	D3	Thesaurus	1.665	D1	Suffix s	1.574	D3	Thesaurus	1.448
3	D2	Stems	1.570	D4	Stat. Phrase	1.566	D5	Concon	1.367
4	D5	Concon	1.559	D5	Concon	1.556	D2	Stems	1.335
5	D1	Suffix s	1.530	D2	Stems	1.534	D2	Binary	1.294
6	D2	Binary	1.494	D2	Binary	1.477	D2	Title only	1.293
7	D2	Overlap	1.455	D2	Title only	1.430	D1	Suffix s	1.283
8	D2	Title only	1.369	D2	Overlap	1.407	D2	Overlap	1.241
Range			0.317			0.172			0.215

Dictionaries used : D1: suffix s; D2: word stem; D3: thesaurus; D4: statistical phrase; D5: word-word association.

Merit: (normalized recall + normalized precision).

Range: Performance range

1. The order of merit is more or less similar in all three collections with minor exception (suffix s performs very well in Cranfield).

2. The performance range of the methods used is smaller for Cranfield collection than for the other two collections.

3. The use of binary weights, overlap correlation and title only is always less effective than the use of weighted terms, cosine correlation, and full document abstracts.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4. The thesaurus method including its synonyms performs better than the word stem or suffix s method.

5. The thesaurus and statistical phrase methods are almost equivalent and most effective.

All these results together indicate that in automatic systems weighted terms should be used, derived from document excerpts and synonym dictionaries should be implemented. Additional improvements which are more tuned towards the local requirements may be gained by adding phrases, hierarchies and term-term association procedures. Phrase languages are not substantially better than synonym thesaurus (which appears to be a surprise). The hierarchy dictionary must increase its resolution in order to be effective. Also automated indexing is not always much inferior to manual indexing.

Finally, in addition to content analysis, it is also necessary to evaluate search methods specially in real-time retrieval systems and the effect of feedback in searches.

2.26 Additional Information services

In addition to information retrieval, there are other information services that are also parts of wider information services; these are some type of text transformations meaning extraction of "significant" sentences from the text of a document, production of index volumes, glossaries, concordances; construction of citation indexes and the use of bibliographic citations for retrieval purposes.

2.26.1 Text Transformation

Text transformation systems may serve a variety of functions, including for example, the reduction in length of a text into an abstract form, paraphrasing of a text or translation of a text from one language to another etc.. In all these situations, the transformation process depends on determining *significant* portions of the text. The text portion being examined may be restricted to individual words, in

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

which case it is necessary to determine word significance or alternatively one may consider the significance of word groups as phrases or even of complete sentences.

The significance of a portion of text depends on the context and there are many different procedures available for its determination. For example, there are statistical methods that use frequency counts and item co-occurrence tests in the text , or there are syntactic criteria based on structural relationships between items (where item can be significant word or significant phrase etc.), or it can be semantic criteria depending on the particular area of discourse or the larger context in which the item is embedded; or it can be pragmatic areas as scene analysis.

First of all consider a number of statistical criteria to be used for the assessment of word significance which are very simple and so plentifully used. Absolute word-frequency statistics is not a popular criterion because this can simply vary with the length of the document. The *relative frequency* is an useful criterion because it compares the frequency of one word with the frequency of other words.

$$f_{wd} = \frac{\text{Number of occurrences of word w in document d}}{\text{Total number of all words in document d}}$$

Now if f_{wd} is computed for many various words over many documents, one may notice that the so called "function words" will display the same constant relative frequency in most of the documents (about 90%). On the other hand, the relative frequency of the content words will vary widely from document to document. Thus it is suggested that a word of significance in a document d can be better measured by comparing its relative frequency in the document d with its relative frequency in the collection of the same subject.

$$\text{So, } r_{wc} = \frac{\text{Number of occurrences of word w in collection c}}{\text{Total number of words in collection c}}$$

The four functions of f_{wd} and r_{wc} can be given as follows:

1. $s_1 = f_{wd} - r_{wc}$

which (for document d) increases with frequency of occurrence of word w and reduces with frequency

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of occurrence in the collection.

2. $s_2 = f_{wd} / r_{wc}$

which varies linearly with frequency of occurrence of word w and also reduces with frequency of occurrence in the collection.

3. $$s_3 = \frac{f_{wd}}{f_{wd} + r_{wc}}$$

Here, the function increases but less faster with f_{wd} and r_{wc} has a reducing effect (but less faster).

4. $s_4 = \log (f_{wd} / r_{wc})$

which is like formula no 2, but the speed is slower.

Since the frequencies f are likely to be large for many words, the first two measures offer the best resolution.

Significance measures as above interesting produce low significance for rare words used infrequently and also for common words used frequently in most documents; at the same time, normally rare words used frequently in a document produce a large significance value. All this happens as expected except the fact that infrequent low frequency words still do not produce high significance value. In certain situation, the significance value of low frequency words can be enhanced by adding some extra weights when they are detected to be of that class. They are easier to detect when they occur in a title, or an abstract than when they appear in the text.

The above definition of word significance can be further extended to *word groups*, or phrases as the basic unit that single words. Such phrases can be identified by dictionary look-up or by co-occurrence statistics of word pairs in the sentences of a document or many documents in the collection. Alternatively, a word group can be defined whenever two or more words with a high significance value occur together (or close enough) in the sentences, or whenever they jointly occur in the same context or when some specified non-significant words come jointly with the significant words. The following cases are of interest I this context:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1. Pairs of significant words joined by certain prepositions or conjunctions as *of, or, and, in* etc..
2. Non-significant words, such as *last, first, more directly* followed by a significant word.
3. Non-significant words such as *name, account, approach, limit* etc. directly following a significant word.
4. Non-significant words such as *case, choice, connection*, which are either preceded or followed by significant word.

Given the significance value of two or more components, the significance of the word-group is given by the product of the significance values of the components. For example, if the two components have significance values v_1 and v_2, then the group will have significance $v_1 . v_2$. If the distance between the components also needs to be taken into account in such way that a growing distance (meaning no of intervening words) will reduce the significance value, then a possible formula for group significance cab. be given as:

$$s = \frac{(c - d)\ v_1 . v_2}{c} \quad \text{for } d < c$$

and

$s = 0$ for $d > c$

where c is some threshold constant of a distance

and d is the word distance.

Here c is the maximum number of words permitted within a word-group. Another possible formula can be say ,

$$s = \frac{1}{2^d} v_1 v_2$$

where d is the number of non-significant intervening words.

Extending exactly the concept of word-group significance to complete sentences, it is also possible to formulate sentence significance based on the significance values of the word-groups in the sentence. The following strategy may be used:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1. The significant words are identified in the text.
2. Those significant words which are further from other significant words by more than c-1 words are dropped from the list (because they are considered isolated).
3. Word clusters are formed consisting of all non-isolated significant words in each sentence; so each cluster starts and ends with a significant word and maximum distance between the components of the pairs is less than or equal to c-1.
4. The significance value s_i of cluster i is calculated by taking the square of p_i, the number of significant words in cluster i, and dividing that by total number q_i of words in the cluster.

   So,

$$s_i = \frac{p_i^2}{q_i}$$

5. The significance value of a sentence is then the value of the cluster contained in it that carries the highest significance number.

In an example sentence " A *concordance* is an *organized index list*"
where the significant words are the italicized ones. This sentence contains a single cluster starting with the significant word concordance, consisting of four significant words and six words. So p_i is 4 and q_i is 6 and so s_i = 16/6. The word significance and word-group significance can serve as the basis for various indexing , abstract generation tasks.

Abstracting systems

Generation of abstracts of a text is a complex task because it must reflect the document content. A typical automatic abstracting system is described with the flow-chart in fig 84. The pre-edit and key-punch incoming text brings in any extra identification information, corrigendum etc.. The next step assigns some document number (any location information if needed) to the document. Before generating abstract, all titles, citations, captions, footnotes etc. needs to be removed leaving only the text for extraction. A frequency count is first made on those words and then the word significance is calculated. In calculation of the significance, the location (i.e. position of the words in the text etc.), the semantics

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

etc. also can be taken into account. Dictionaries of common words and important words (giving a restriction, selection and synonym list) with some syntactic and semantic markers will enhance this function. A sentence significance score is then assigned using as a basis individual words and word groups. The sentences are arranged in the decreasing order of the significance. A cutoff value (n) is chosen and top n sentences are selected for inclusion. These sentences are arranged in the text order. This extract is then printed out along with the bibliographic information.

Clearly these abstracts may not be ideal, because the normal flow of ideas is interrupted; also even if the sentences may reflect the principal ideas, the sentences containing follow-up reasons or the guiding context are often absent. Also these extracts mostly do not contain any subjective comments, editorial remarks, concluding message etc..

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

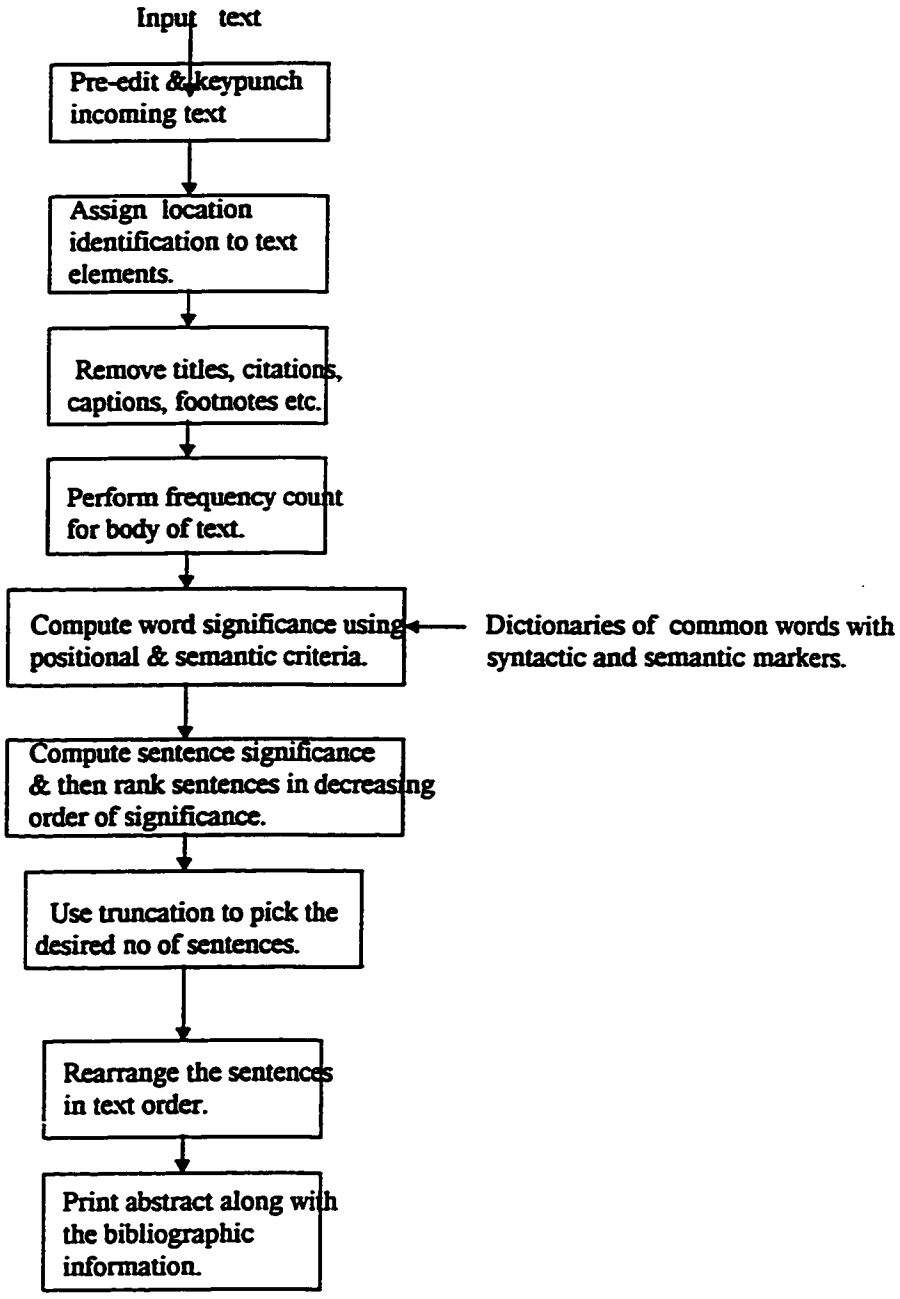


Figure 84:Typical automatic extraction system

There are many methods available for producing better automated abstracts, such as using synonym dictionaries and thesauruses replacing some words with more effective synonyms. Also many rejection criteria & rules can be used for rejecting some words from the document text. Also syntactic

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and semantic factors can be included to add or change some constructs as part of the abstracting system. It has been suggested that generative grammar be used to generate sentences following certain syntactic and semantic rules thus replacing certain sentences originally in the document text.

Generally speaking, one supplies a source text, together with word class assignments for the individual word types needed for sentence transformation. A syntactic analysis of the extracted sentences is performed first and then the grammatical relationships between the significant words is analyzed. The generation of new sentences proceeds next, starting with selection of a sentence type and then using dependency or phrase structure grammar to build a syntactic frame for the sentence. The actual output is limited to a coherent discourse by choosing the elements of the terminal vocabulary from the source text and using the similar grammatical relationship between the significant words as in the original sentence. Also it is suggested that a transformational grammar may be used to create better and more complete abstracts.

2.26.2 Index generation

An index is a list of items together with the item identifiers. For example, an index can be a list of keywords in a document text along with the line numbers where the keywords are used in a document. Indexes can have different characteristics. For example, it can be *term oriented indexes* where individual keywords are listed with document identifiers. Also, it can be *document oriented indexes* such as book catalogs, citation indexes or bibliographies etc.. A term index offers some indication to the document text, whereas document oriented index is an indicator of library holdings or a directory pointing to a selected class of documents.

Within the set of term indexes, sometimes called concordances, it is possible to make a distinction between text word indexes, based on the words in a given text, and indexes of special terms, such as keywords or index terms which may be manually assigned or generated through a very complex method. The same distinction is meant while talking about *derived* indexing, which means indexing derived from the text of a document. There is another type of indexing called *assigned* indexing where

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

indexes are assigned from a schedule of descriptors. Of course, in a fully automated indexing system such distinctions are blurred. A similar classification in term index types is made using classes of *indicative* indexes as opposed to *informative* indexes. The former gives a superficial classification of the subject area whereas the latter names the core subject matter of the document in the most specific way.

Term oriented indexing

A full concordance is a fully ordered list of entries, for example, words used in a document collection, one entry used for each word type. Each entry is composed of the entry type and then a list of citations for each occurrence of the item in the documents. As a simple example, the keyword index can be in the following format:

concept number, Document number, weight

which is sorted in the order of concept number and then document number. The weight is the term weight assigned to the document.

As enhancement, it may also include the location (the sentence #) for each occurrence of the term. There are many different formats for the term indexes. In addition, there are variations for concordances from a full concordance. A *selective* concordance is one containing entries for those terms contained in a list. A *restricted* concordance is one containing all entries except for the ones in a list (exception list). The permuted title index (also known as KWIC) is a very popular index list because it gives quick access to the document from the significant words in the title.

To produce a KWIC index, the important sentences or phrases are chosen from the text. These excerpts are sorted in order of significant words. Sometimes a simple syntactic analysis may be needed to eliminate unnecessary parts of a title.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3. HIGH-SPEED NETWORKING & CELLULAR NETWORKS

3.1 Networks

Networks, being the means of communications can be classified in various ways. For example, from ownership viewpoint, there are private networks, public networks, semipublic networks as found in the universities. From medium viewpoint, there are twisted pair cable networks, coaxial cable networks, Microwave networks, Radio networks, Satellite networks, fiber optic cable networks etc. Fig 85 shows a taxonomy of such a rough classification. Then there are digital networks and analog networks. From application point of view, there are data networks, voice networks, networks combining data, voice, video, computer graphics, cellular networks etc..

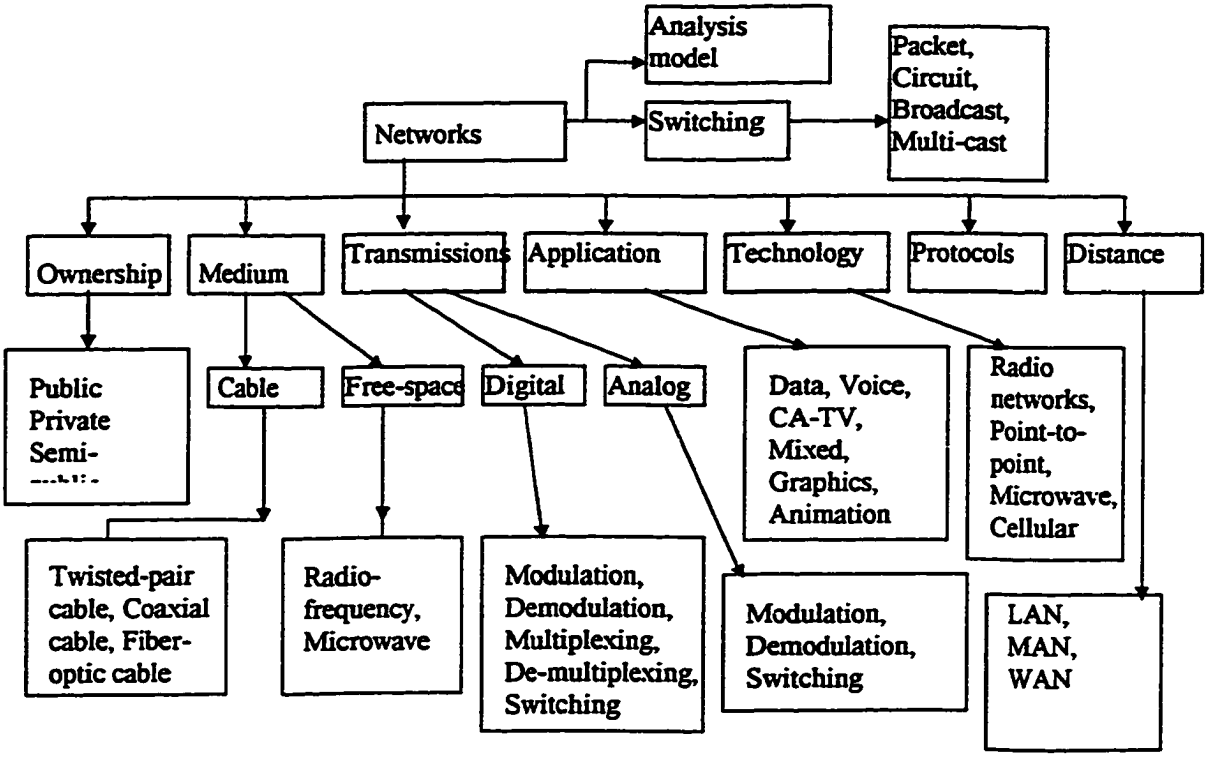


Figure 85:Taxonomy of networks classifications

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

These networks are vast all over the world and many times they are inter-connected through internetworking, they follow various standards and protocols using many kinds of equipments, based upon different network models. The various aspects of networks are shown in fig 85 although some have not been elaborated due to lack of space. This field is a large universe covering areas as medium characteristics, transmissions types, technology, analysis models, topology, devices etc.. Of late, many new technologies have appeared in the horizon. They are:

1) High-speed networking
2) Cellular network
3) Information network.

3.2 Introduction to high-speed networking

With the rise of supercomputers, advanced processor chips as DEC Alpha, Power/3, UltraSparcII, Intel's Merced etc. , multiprocessor systems, cheap fiber-optic links high-speed networking has become the state of the art today. Currently there has been tremendous explosion in network bandwidth. Today local area networks run at the speed of 1 Gb/s and wide area networks are not far behind. As an example, the main superhighway part of Internet today uses very high-speed backbone network service of NSFNET with IP/ATM(AAL5) on OC-12 (using 622 Mb/s) and IP version 6 with 1.244 Gb/s speed network as the domestic superhighway for future Internet is well on its way. Also demand for data communication has risen substantially over the telephone network suggesting to redesign of network to support digital data. Also it is commonly felt that reengineering opens up the possibility for on-demand video. In general more demand in communications, have increased the need for high speed communications raising the limits to gigabits per second. E-mails worldwide, distributed file system, distributed computer processing, video and audio-conferencing, exchanging files, multi-casts, virtual reality etc. have added the need of faster data communication.

One powerful idea in data communications has been internetworking. The open-ended protocol of internetworking is to ensure movement of packets of data over many heterogeneous networks multiplies

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the possibility of data sharing. The development of the worldwide Internet, a global network connecting millions of networks and hosts using the TCP/IP protocol suite is an example of the power of internetworking. The telecommunication networks are carrying more data. Also telephone companies are trying to consolidate cable television and telephone service in one integrated services network. Also today's telephone networks are mostly digital today. Most of the cables in America have changed from copper to fiber. As a result, there is a surplus of bandwidth, a change in the type of traffic and commercial reason to change every network into a unified network.

These interests from different communities converged into joint research activity of creation of gigabit testbeds where researchers from various industries collaborated to find viable schemes for high-speed data communication. One of the results was standardization of Asynchronous Transfer Mode (ATM), a suite of communications protocols designed to support integrated voice and data networks. ATM - a new protocol designed for high speed communication has been again designed on the basis of a layered architecture. A layered architecture always delineates the responsibilities of the various protocols clearly and that is why the standards committees always specify the protocols using the layers. For example, OSI network protocol is based on a seven-layer model and TCP/IP is based on a five layer model and these layered architectures are described schematically below in fig 86.

Development of optical fibers have taken place over the last two decades adding much greater bandwidth to the available networks. Today all the public networks in America and Japan are on fibers. Many other countries are also replacing their copper trunks with the optical fibers slowly. The transmission technologies of fiber can support transmission and reception at gigabit rates while ensuring a high quality of communication at a very cheap price. It operates using the principles of reflection and refraction of light. The medium carries pulses of beams of lights from LASER sources through optical amplifiers upto the optical detectors. Light travels at 0.69 times the speed of light (i.e. 2.1×10^{8} m/s) through glass fibers. The transmitted signal sits as a modulation signal upon the light carrier thus providing high signal bandwidth (even terabits). Thus a hair-thin strip of fiber can carry thousands of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

multiplexed signals at gigabit rate. A strand of glass called core is wrapped by higher density glass called cladding where the outer cylinder is protected by a protective covering.

In addition, there are other types of networks such as packet radio networks, Microwave networks, satellite networks which will also support high-speed gigabit data transfers. Packet radio networks can transfer data among the ground stations using FM, satellite networks use TDM as the transmission mode. The satellite networks can use analog transmission or digital transmission using transponders. Normally they operate with microwave frequencies in the band 2-40 GHZ although the band 1-10 GHZ is most ideal for communication. One satellite splits its 500 MHZ over a dozen transponders giving each about 40 MHZ bandwidth. The satellites have an uplink channel for receiving from ground (or another satellite) microwave link and a downlink channel for transmitting down to a ground station communicating using dish-shaped antenna. Satellite links are used for television distribution, long distance telephony, private business networks etc.. Microwave networks are almost like radio networks except for the fact that radio networks are omnidirectional and microwave networks are focused. Terrestrial microwave use parabolic dish-antenna of size about 10 ft in diameter located on tall towers communicating on the basis of line-of-sight. Their frequency bands are approximately 2-18 GHZ with bandwidth upto 220 MHZ carrying digital data or analog signals. Packet radio system uses ground based antennas to link multiple sites in a data transmission network. Radio transmission covers VHF and part of UHF bands: 30 MHZ to 1 GHZ.

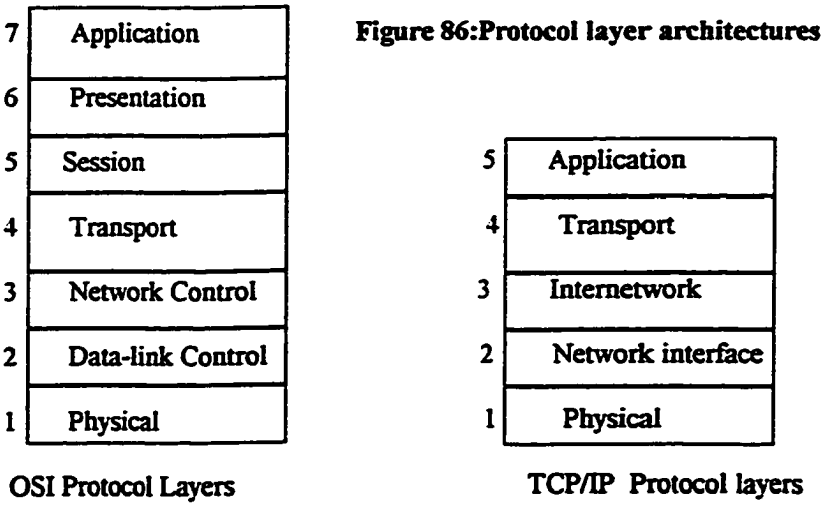


Figure 86:Protocol layer architectures

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Finally, the introduction to high-speed networking can not complete without mentioning about ISDN which is an intermediate step to high-speed networking. Integrated Services Digital Network (ISDN) is a design for a fast and digital network that will integrate voice and data in one cable. This is a CCITT recommended solution for a smooth phase-out of the analog telephone systems which will offer lots of additional values on top of the POTS. Some examples are conference calls, automated data display, call forwarding, database access, home alarm, fax, telemetry transmission on the same cable. Its new features are CCIS (Common Channel Interoffice Signaling), digital user interface including digital PBX, channel splitting and support for multiple channels of communication. But very soon, the speed limits (64 Kb/s on each channel) became a difficult issue and it will not suit the need for computer-based communication needing very high bandwidth. Also the fixed-bandwidth style of ISDN does not suit all applications. As a result, CCITT has recommended broadband ISDN (B-ISDN) raising the bandwidth requirements which will provide many broadband services as well as ISDN services.

3.3 Optical fiber

Optical fiber has become a very popular media today for its near lossless and high bandwidth capacity which carries optical beams through its thin (2-125μm) strands. A strand of high-density glass at the center called the core, is surrounded within another low-density glass layer called cladding as shown in fig 87.

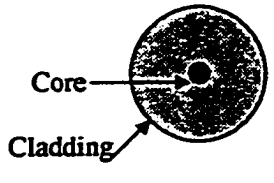


Figure 87:Cross-sectional view of a fiber

The core has higher refractive index and the cladding has lower refractive index. As a result, any light traversing through the core will tend to remain within core because most of the light that tends to escape

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

will reflected back into the core (total internal reflection). In effect the fiber acts as a waveguide for frequencies in the range 10^{14} to 10^{15} Hz which covers infrared spectrum and part of the visible spectrum.

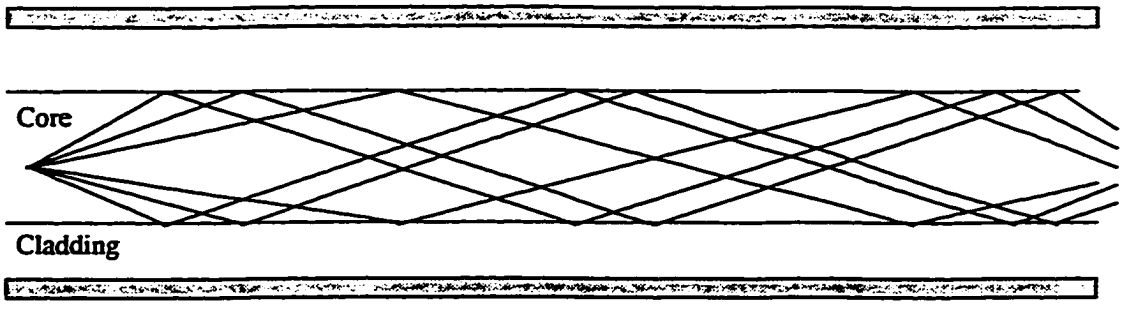


Figure 88:Beam propagation in multimode fiber

Fig 88 illustrates the light propagation in multi-mode optical fiber. Light is emitted from the source in the core either from LED or from injection Laser diode. Rays at shallow angles are reflected and are propagated along the core; other rays are absorbed by the cladding. Such propagation is called multimode where there are beams at many angles that will reflect; by reducing the radius of the core down to order of wavelength, only a single angle can pass and all others are absorbed thus producing an axial ray. Single-mode fibers cause little dispersion at the receiver and thus supports higher transmission rate whereas multi-mode fibers communicate slower. But multi-mode fibers are less expensive and they fit the transmitter or the receiver attachments more easily and that is why they are used so commonly. Chromatic dispersion is another kind of dispersion that happens at the source (it is impossible to generate light at a one pure wavelength). Material dispersion is the result of impurities in the fiber which cause photons to travel at varying rates. At the receiver, photodiodes are used which may be PIN type or APD type. PIN photodiodes have a P layer and an N layer of a diode with silicon in between. The APD, avalanche photodiode uses a stronger electric field, but both are basically photon counters.

The amplitude-shift keying technique is used to transmit digital data over the optical fiber. The light frequencies used are 0.85, 1.3 and 1.5 microns each of which bands has about 25 Thz. Standard

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

signaling devices can signal 1 to 1.4 bits per Hz and so a single fiber can transmit about 30 terabits per second. The frequency capacity of a band is given as

$$\nabla f = c \nabla\lambda / \lambda^2$$

where $\nabla\lambda$ is the bandwidth of the wavelength (about 200 nm), ∇f is the band-width in frequency, λ is the center wavelength (1.5 μ) and c is the speed of light (2.1×10^8 m/s). In reality, the achievable rate is less. Building fast signaling device is costly; it is more cost-effective to build parallel signaling devices that transmit at different wavelengths at the same time with some gap between the wavelengths so that they do not interfere with each other. All factors considered, a single fiber can be used to transmit at 10 Tb/s speed.

For using fiber optic components in a network, the default signaling protocol with fiber is SONET (synchronous optical network) and probably it will become very popular in future. SONET is a part of a larger suite of telephony standards called *Synchronous Digital Hierarchy* (SDH). SDH gives a standard multiplexing rates for high speeds. Table 3-1 provides the hierarchy of SONET/SDH transmission rates.

Table 3-1: SONET/SDH transmission rates

Data rate (Mb/s)	SONET STS/OC Designation
51.84	OC-1
155.52	OC-3
622.08	OC-12
1,244.16	OC-24
2,488.32	OC-48

Under SONET standard the first data rate is 51.84 Mb/s which is *Synchronous transport signal* level 1 (STS-1) or Optical-Carrier 1 (OC-1). The subtle difference between STS-n and OC-n is that OC-n describes an STS-n bit stream after it has been converted into optical signals. The numbering under STS/OC scheme indicates how many STS/OC-1 circuits can be encapsulated at the given rate. For example, STS-12 at 622.08 Mb/s equals 12 STS-1 circuits. So an STS-12 line can be used as 12 multiplexed STS-1 circuits or as a single 622.04 Mb/s line. The CCITT recommended hierarchy is *Synchronous Transport Module* (STM-1). SONET transmits data in frames which are made of 90

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

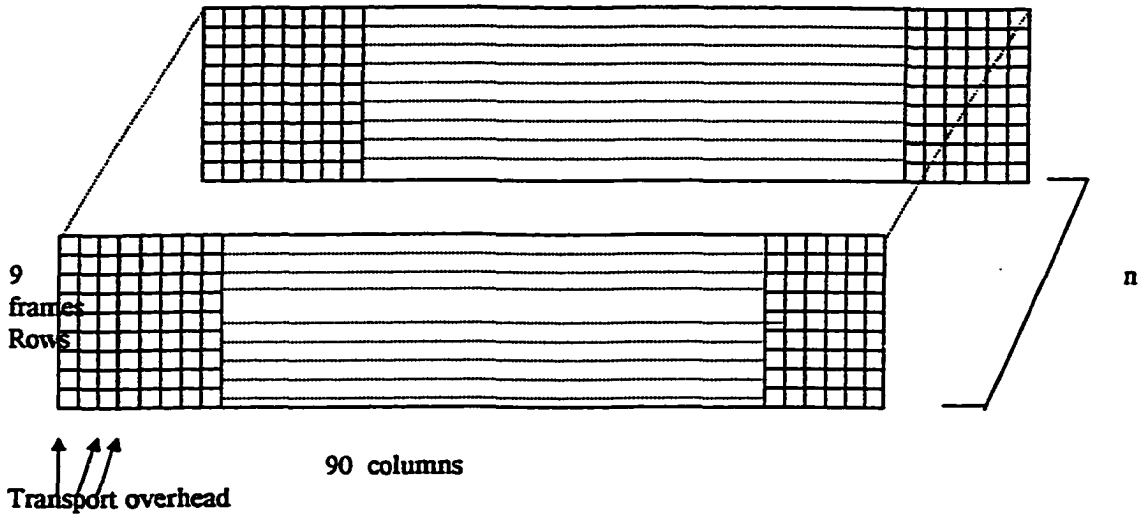


Figure 89: Logical view of SONET frames

The first 3 columns (i.e. 27 bytes per OC-1 frame) are reserved for transport overhead information. The first three rows are called the *regenerator section overhead* and the last five rows are called the *multiplexor section overhead.* Regenerator overhead is processed at each SONET repeater. Multiplexor overhead is processed at SONET multiplexors. The fourth row simply contains pointers. A key problem in multiplexing is that the lines attached to a multiplexor typically get out of sync with each other. Each line has its own clocking mechanism to manage its transmission and reception of bits, and they tend to differ very soon. The pointers are used to adapt to these variations. SONET permits data to start arbitrarily anywhere in the payload and these pointers indicate where in the payload the data starts. In addition, one column in each 90×9 frame may be used for placing multiplexing information within the data (called *path overhead).* If this path overhead appears only in the first 90×9 frame, then the SONET channel is concatenated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

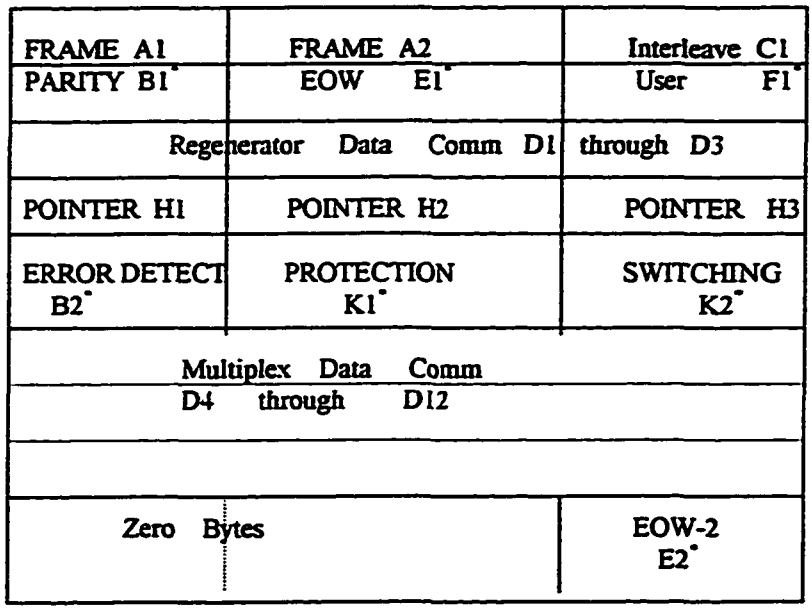


Figure 90:SONET transport overhead bytes

Fig 90 illustrates the structure of the three columns of SONET transport overhead information for OC-1. At higher rates, the starred (*) bytes appear only in the first 90×9 frame; the other bytes may be repeated in later frames. The A1 and A2 bytes form a frame alignment word, used for synchronization of the start of each 90×9 block. The C1 byte is used for interleaving STS frames (at rates above OC-3). The B1 and B2 bytes are error checks on the regenerator section overhead and the multiplexor section overhead respectively. The E1 and E2 bytes are used to provide a 64 Kb/s voice channel for maintenance personnel in each of the regenerator and multiplexor sections. The F1 byte is a 64 Kb/s alarm channel. Bytes D1 through D3, and D4 through D12 provide data communications channel for network management between regenerators and multiplexors. The data communications channels are of interest because they are used to signal between SONET equipment. Using these channels a user can instruct multiplexors to change routing of SONET channels, making it possible to build a switched SONET network in which connectivity between different nodes of a SONET network is under user control. Sometimes the format of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the overhead bytes may differ between two different vendors. For the data portion, standards exist for carrying multiplexed telephony circuits (at certain SONET rates) and for carrying data in ATM cells.

In contrast to SONET, Wavelength Division Multiplexing (WDM) networks use principle of wavelength division multiplexing (communicating on many wavelengths at the same time) by employing multiple electro-optical devices in parallel. There are two key challenges in designing WDM networks. One challenge is to minimize the amount of time spent deciding which systems will communicate on which channels and thus maximize the data transmission time. The second challenge is to make effective use of the fiber bandwidth, both by packing the different frequency channels as close together as possible and by improving the effective tuning ranges of optical devices. There are two main types of WDM networks. 1) single-hop where all nodes can communicate directly with all other nodes. 2) multi-hop WDM networks where a node can only communicate with a few nodes and so to reach other nodes the message has to be relayed through another connected node.

3.4 Asynchronous Transfer Mode

ATM is a form of packet transmission across the user-network interface in the same way as X.25 is a form of packet transfer. The protocol model for ATM is illustrated in fig 91.

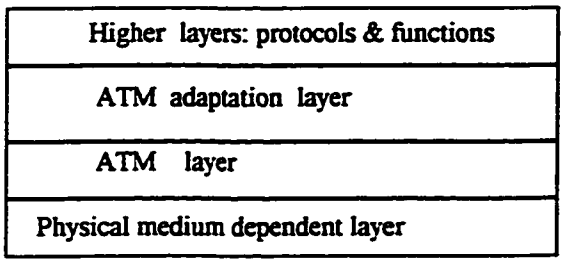


Figure 91:B-ISDN protocol model for ATM

The ATM layer uses small sized packets called cells (5 Bytes header + 48 Bytes data) each of which is 53 Bytes long and the network will be a packet-based network connected using ATM switches. The small cells can support circuit-switching applications also. The various types of adaptation layers makes it

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

possible to use from various types of applications. The signaling is done separately using common channel signaling and some of the other features of ATM are:

- No link-by-link error control
- No link-by-link flow control
- End-to-end error control if required
- Use of internal virtual circuits
- Switching based on table lookup;
- Fixed packet size

Cells move through the switches in the network by making hops from switch to switch. Cell routing through the cell network is done using the method of source routing or hop-by-hop addressing schemes.

Source routing: In this scheme, the cell's full route is annexed in each cell as a sequence of route elements, each element being the output port number of a switch. When the cell enters a switch, the switch looks at the first element of the source route and sends the cell to the respective port after removing the first element from the source route. The limitation of it is no. of hops for a cell in the network is limited, because the 5-Byte header can not support many port numbers.

Hop-by-hop routing: This scheme uses a fixed-size header containing a hop-id. When a cell enters a switch, the switch looks up the hop-id in a table. The routing table contains 3 columns: the hop id if the inbound cell, the output port the cell should exit through and the new hop-id to replace the existing hop-id. Fig 92 gives an example of hop-routing. A cell with hop-id 27 enters port 1 and the exiting cell goes on output port number 6 with new hop-id 94. Also there are other variations on hop id scheme. One variation of it has a single hop-id that is carried by the cell on all hops. Another variation is hierarchical hop ids, where some part of the network routes according to one part of the hop-id, another section of the network routes according to another part of the hop-id. For example, ATM networks use a two-level

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

hierarchy that divides the hop-id into a path-id (for routing a group of cells together) and a circuit-id (which routes individual cells).

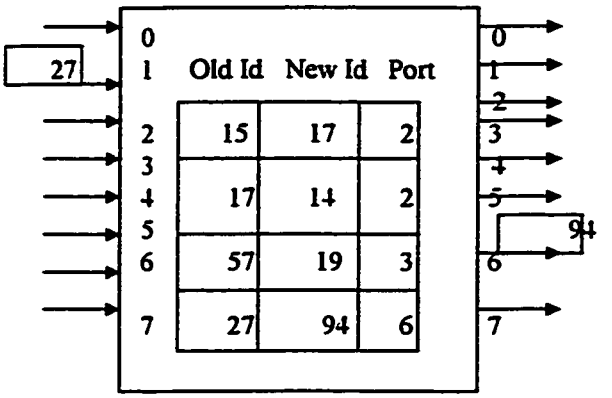


Figure 92:Hop routing in Cell networks

3.4.1 Adaptation Layer protocols

It is generally assumed that a cell protocol will be used through an adaptation layer as in fig 91. The purpose of this adaptation layer is to make the cell services more adapted to the higher layers or the applications using them. One reason that adaptation protocols are needed is to aggregate cells into larger data units. Applications tend to transmit data in large amounts which is much bigger than a cell (say as file blocks or video frames etc.). Hence these large units need to be assembled into cells and again reassembled into large units by the receiver. This protocol is called segmentation and re-assembly (SAR) protocol. There are various functionalities available from the Adaptation layer, but the adaptations layer overhead takes away precious space from the small 48 bytes of data.

Regarding errors, there can be two types of errors in cell networks: 1) cells missing, 2) bit-errors in cells. In the first case, missing cells can be recovered by adding extra recovery cells in the input stream. One possible scheme is XOR-ing 4 consecutive cells and adding this extra cell after those cells. But if more than one cell is lost, this scheme fails to recover. A more sophisticated version may be to arrange the cells in a square block and compute XOR of the cells along the rows, and also compute XOR of the bit-diagonals. Both the types of XOR cells will be added in the stream. Such a scheme can recover upto 2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

subsequently lost cells. Another recovery scheme uses a simplified Reed-Solomon code, called Reed-Solomon erasure (RSE) code. This is a popular encoding technique that sends h check bits along with every k bits of data. The h check bits along with the data in k bits are used to build h linearly independent equations. At the time of transmitting, the h check bits are computed from the data such that the equations all equal 0. Upon receiving, the equations are checked to see if they equal 0 and if not, the equations are solved to find the position whose value is in error and its correct value.

For all these methods, there is no need to buffer cells prior to transmission while computing the recovery codes. But if a cell is lost, the receiver will need to buffer up some cells to recover the lost cell. In the second case (i.e., in cases of bit-errors within cells), error correcting Crocus may be added in the cells. The number of bits in the error correcting Crocus is a bit too large for the small size of cells (because the number of bits is proportional to the log of the size) and thus considered an overhead. For example, some ATM SAR protocols use a 10-bit CRC to protect 48-byte payload, carrying an overhead of 2.5%.

ATM Conceptual Model

The conceptual model for the design of ATM is based on some principles. First, ATM is designed to work in a hierarchical network much like the telephone network with the local user equipment connected to regional ATM providers, who are in turn connected with national or international ATM providers. ATM users' equipment connects to networks via a user-network interface or UNI. This interface supports multiplexing and protects ATM networks from misbehaved user equipment. But connections between provider's networks are made using network-network interface or NNI.

Secondly, ATM provides a connection-oriented service- i.e. a cell may not be transferred until a channel is established. Normally the ATM channel identifiers are kept small (28 bits). Because ATM is hierarchical, the channel identifiers also have a hierarchical structure. The actual channel identifier is made of two components: a channel identifier and a path identifier. At the ends of transmission where links may be fanning out to the users, cell routing takes place using both the components. But the cells traverse through a backbone network that is made of high-speed trunk links connecting each switch to its

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

neighbors. In these switches it is only necessary to determine which link to route a cell to. Because these links are few, only the path identifier is enough to decide the next hop.

Another principle of ATM architecture is to minimize loss and errors. Originally it was believed that ATM networks should strive to keep the loss rate low as in fiber (1 bit error in 10^{12} bits or better). But it has been found that occasional surges in traffic in ATM networks may cause a switch to get overloaded and drop a burst of cells. To keep such burst loss under control, ATM ranks cells and the expectation is that high-priority cells will still have low loss rates as in fiber. Also to keep ATM user equipments within low cost, cell processing has been made easy by avoiding cell reordering. As cells are in order, the user equipment may not even need any buffer. So ATM telephone can just play the voice samples that arrive in the cells.

3.4.2 ATM Cell Format

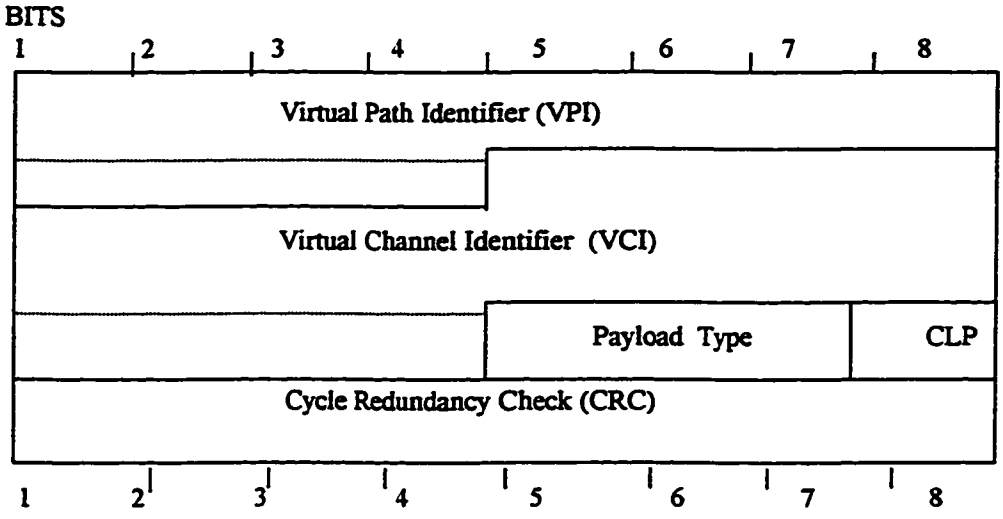


Figure 93:ATM Header at Network-Network interface (NNI)

ATM cells are 53 bytes long, where the first 5 bytes (cell header) are used by UNIs, NNIs and switches to route the cell. The remaining 48 bytes form the data formatted in an ATM adaptation layer. But the structure of the cell header is different for UNI and NNI.

NNI format

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The NNI format of the cell header has six fields as shown in fig 93. The first two fields in the NNI header are the 12-bit virtual path identifier (VPI) and the 16-bit virtual circuit identifier (VCI). Each ATM connection is uniquely identified by the combination of these two fields making a 28-bit address. The VPI and VCI provide a two-level addressing and routing hierarchy. ATM switches route based on both VPI and VCI or just on VPI. The VPI and VCI fields are small because they are the indices into the routing tables and there is a high chance that two connections may have the same VPI and VCI in the same network. One solution to the problem of making it unique is to make them unique at each hop (on a hop-by-hop basis) as described above in hop routing. For each link, the two switches at each end agree on the VPI and VCI pair to serve as the hop id for each channel. When a cell passes through an intermediate switch, it changes the VPI-VCI pair to a new pair for the next hop. Switches that use only VPI, change only the VPI and retain the VCI. This scheme will avoid collision because each pair of switches knows which VPI and VCI pairs are in use on a given link and will not assign the same. Payload-type field is a 3-bit long field used to distinguish between user traffic and various forms of operations, administration and management (OAM) traffic as in table 3-2.

Table 3-2: OAM Payload Type Values

Bit Pattern	Definition
1 0 0	OAM link associated cell.
1 0 1	OAM end-to-end associated cell
1 1 0	OAM resource management cell
1 1 1	Reserved for future use

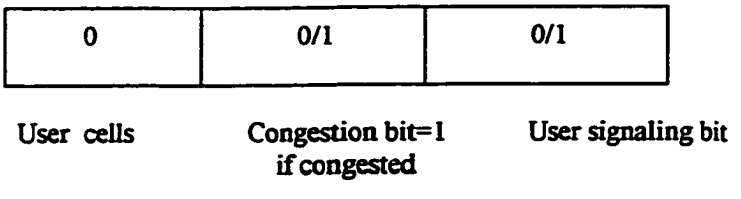


Figure 94:Payload field format for user data cells

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

If the first bit in the field is 0, then the cell is a user data cell - moving from one user to another user. The organization of the payload-type field is shown in fig 94. In the user data cells, the second bit is used to indicate if the cell encountered congestion or not. The signaling bit is used by one of the adaptation layers (AAL5) to mark the end of each datagram. If the first bit is set, then the cell is some form of management cell. The cell loss priority (CLP) is a single bit indicating to a switch, if per mischance the switch should drop some cells, this cell can be dropped before any cell that has the CLP bit clear. This bit supports two levels of traffic priority and also gives some protection against abusive users. ATM commits a guaranteed bandwidth to a connection. If the user sends data at higher rate, the ATM standard permits the network to set the CLP bit in as many cells as necessary. The CRC is 8-bit long computed over the 5-byte cell header. As the VCI, VPI, congestion bit may change at each hop, the CRC must be checked and computed at each hop. The CRC polynomial is $x^8 + x^2 + x + 1$ which can correct a single-bit error and detect a large class of multiple-bit errors.

UNI format

The ATM header format in the UNI is slightly different from the header in NNI. The UNI header format is shown in fig 95. The major differences are that part of the VPI field has been replaced with a generic flow control field (GFC) and that the number of VPIs and VCIs that can be active at a UNI has been limited because of the bits taken up by GFC.

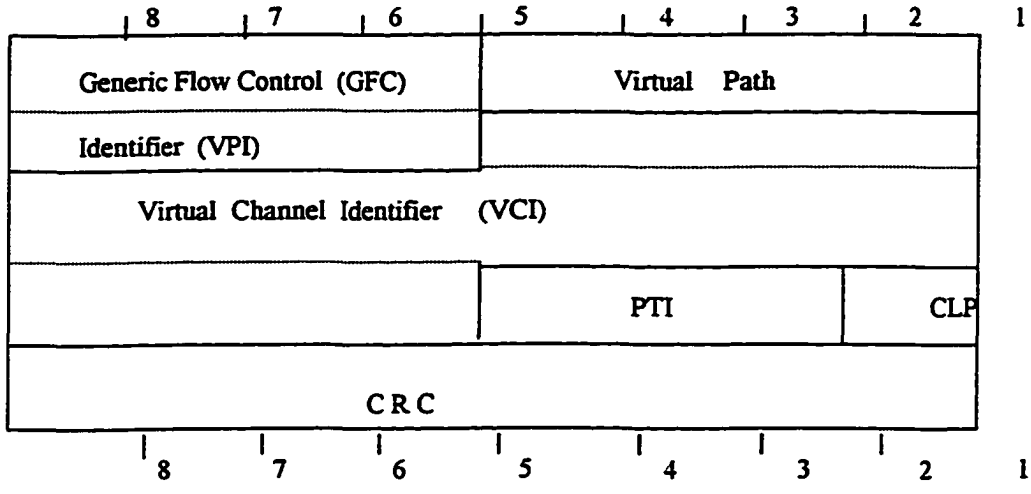


Figure 95:ATM header at User-Network interface (UNI)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Users may connect the ATM network with shared access networks such as a Distributed Queue Dual Bus (DQDB) ring. The purpose of GFC field is to give UNI a four-bit field to negotiate with the shared access networks about how to multiplex the shared network among the cells of the various ATM connections. The values of the GFC fields are currently not defined, and the value of zero is currently used. But there are many controversies about the real utility of the GFC field due to which the number of possible VCI and VPI values have been further reduced. So ATM is not a *concatenative* protocol due to its two interfaces and there is always a boundary between the users and providers in the network.

3.4.3 Design of Adaptation Layer

The purpose of an adaptation layer is to package the higher level of data from applications such as datagram, voice samples, video frames into series of cells to be sent over ATM connections. The CCITT took a multi-choice approach to designing the ATM adaptation layer and designed a suite of adaptation layer protocols with different layers for about four different types of applications.

1. Constant bit-rate applications. These applications send and receive data at constant bit rates. They require a bounded delay between the source and destination. Examples are telephones and some present data video systems. Also a constant-bit rate service can replace a leased line. ATM Adaptation layer 1 (AAL 1) protocol supports such service.
2. Variable bit-rate applications. These applications send data at variable bit-rates but still require bounded delay. AAL 2 provides such service.
3. Connection-oriented data applications. This class was intended to support applications using network services as X.25.
4. Connectionless data applications. Those applications use datagram networking protocols such as UDP/IP or TCP4/CLNP.

The initial protocol format AAL 3 and 4 were so similar, that they were merged to call AAL 3 / 4. At the same time another simple and efficient adaptation layer (SEAL) has been designed and it is called AAL 5 and also AAL2 has been dropped.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The CCITT recommended to break down each AAL into two layers: a segmentation and reassembly (SAR) sublayer, responsible for breaking data into cells and a convergence sublayer to manage the flow of data to and from the SAR sublayer.

AAL 1 The AAL 1 has two component layers as SAR and convergence layers.

AAL 1 SAR

AAL 1 SAR formatting uses one byte (among the 48 bytes) at the start of each cell. This bytes has four fields: a one-bit convergence sublayer indicator (CSI), a three-bit sequence count (SC), a three-bit CRC, and an even parity bit. As a confusion, sometimes the first nibble is also called the sequence number (SN) as it contains a sequence count and the second nibble is called sequence number protection (SNP) field. The CSI bit is used for signaling purpose and currently it is used to indicate boundaries of error correcting blocks in the convergence layer. The sequence count I a counter modulo 8, incremented once for each cell. It is used to detect a cell loss. The CRC is calculated over the first nibble using polynomial $x^3 + x + 1$. After the CRC is computed, the parity bit is calculated over seven bits and placed in the eighth bit. At the receiver, this bit is used to detect errors.

AAL 1 Convergence Layer

The AAL 1 allows users to divide a high bandwidth transmission network into a few smaller fixed bit-rate ATM circuits. Currently there are many designs of this layer one of which is relevant to high bandwidth communications. The convergence layer breaks the video data into blocks of 128 cells. The blocks are arranged as in fig 96.

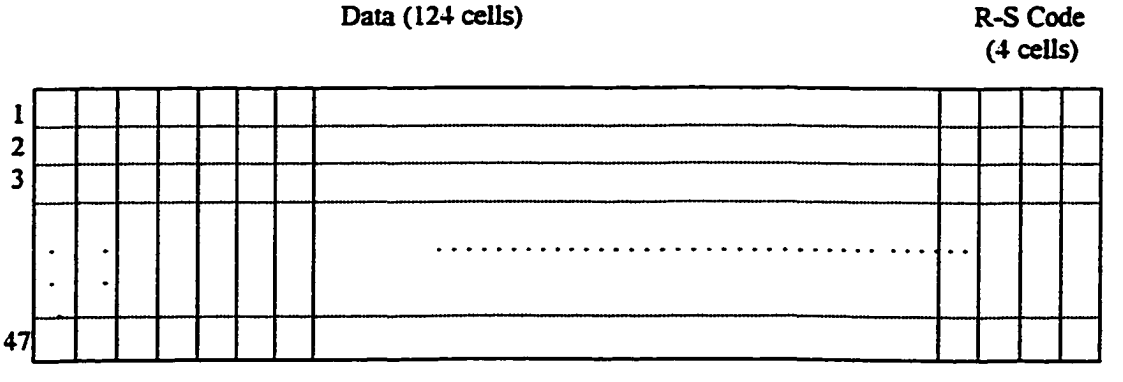


Figure 96:AAL 1 Block for video

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Each vertical column is a cell and each row, r of the block is made of the r-th bytes of all the cells. The last four bytes in each row are a Reed-Solomon code, which can correct upto two bytes in error or recover upto four bytes that have been lost among the 124 bytes in the row. So in a block, this R-S codes can recover upto four lost cells.

AAL 3/4

This is a merge of two standards where AAL 3 was intended for framing service for connection-oriented data protocols like X.25 and AAL 4 was for a framing service of Connectionless protocols like IP. But it is believed that AAL 3 /4 is not suited for general computer-to-computer communication and hence AAL 5 evolved.

AAL 3 /4 Segmentation and Reassembly Layer

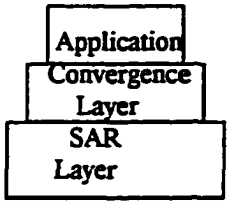


Figure 97:Breakdown of the adaptation layer

Fig 97 shows a breakdown of the adaptation layer where the Segmentation and reassembly layer takes the packets from the convergence layer and builds cells for transmission. The SAR format for AAL 3 /4 is shown in fig 98 showing that SAR layer takes 2 bytes of header and 2 bytes of trailer in a cell for its function (i.e. packet segmentation and reassembly) thus leaving 44 bytes for the data.

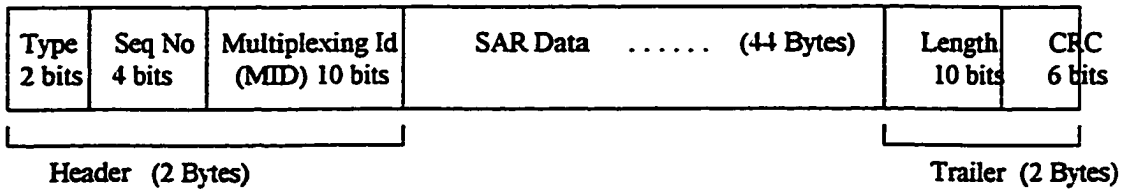


Figure 98:AAL 3 /4 SAR format

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The type field indicates if the SAR cell is at the beginning, middle, end of a packet or a single cell containing the whole packet. The types codes are in table 3-3.

Table 3-3: AAL 3 /4 SAR Types

Value	Definition
1 0	Beginning of Message
0 0	Continuation of Message
0 1	End of message
1 1	Single segment message

The 10-bit multiplexing identifier (also called MID) permits interleaved multiplexing of multiple packets on the same ATM connection. If no multiplexing is done, the MID is cleared. If packets are interleaved, each packet will have a different MID, so that the receiver can distinguish between the cells of different packets and rebuild the packets without mixing them. Such multiplexing may not work very well with applications at times, specially when the number of applications is not very large. In such cases, it is an extra overhead. The 4-bits sequence number works similar as in AAL 1 and counts sequential cells within a packet modulo 16 (i.e. within a MID). The six-bits length field counts the number of bytes of data field. The length field must be 44 except for in SSM (single segment message) and EOM (end of message) cells. In EOM cells, this number can be 63 which means the packet is being aborted (and so should be discarded). The 10-bit CRC can correct a single One-bit error in the cell using the polynomial $x^{10}+x^9+x^5+x^4+x+1$.

AAL 3 /4 Convergence Layer

The AAL convergence layer defines the format for all packets to be sent using the AAL 3 /4 SAR layer and its format is presented in fig 99.

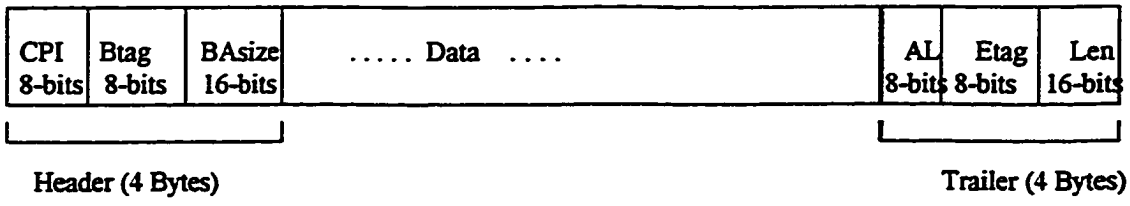


Figure 99: AAL 3 /4 Convergence Format

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The headers and trailers must be aligned to 32-bit boundary and so the trailer may have a padding varying upto 3-bytes long. The common part indicator (CPI) is an eight-bit field which permits the convergence layer header to be redefined for a connection. Currently this field is zero and so the header fields are formatted as follows. But in future further values can be used and it will accompany further definition of the header fields.

The 16-bit buffer allocation size (BAsize) field indicates the maximum number of bytes required to store the convergence layer packet at the receiver. It is expected that some systems may start fragmenting and sending a packet down the connection before all the data has been passed down from the application which is known as *streaming*. In such cases the BAsize field can be used to tell the receiver the buffer size to allocate for containing the incoming packet. The eight-bit beginning and end tag (Btag and Etag) fields are matching values used to let the receiver verify that the header and trailer for a convergence layer packet refer to the same packet. These fields repair a defect in AAL 3 /4 SAR. It was explained before that SAR sequence number is only four-bits long and may restart on each packet. Now suppose all the packets on a connection are 2 cells long each and the seq number starts with 0 for each new packet. Now suppose a burst of error causes two adjacent cells (one, end of the packet and another beginning of a new packet) to be lost. So the receiver gets an end-of-packet cell and then a beginning of a packet cell. The seq no will be fine and so it will be accepted by the SAR layer. Such errors are avoided by the Btag and Etag flags. An assembled packet will have different Btag and Etag at the convergence layer and thus error recovery will begin. The alignment byte (AL) is 8 bits zeroes stream to make the trailer 32-bits long. The length field is 16-bits long. The length field may be redundant considering that the length field in SAR cells can be added up to calculate the length of a packet.

When it applies to data communication, it appears that checking up the segment types is required for each cell and thus each cell needs to be processed as it arrives. This requirement makes processing by the computer inconvenient and so interface cards have been created that can take care of such cell level processing without involving the processor. Also the header-trailer large overhead (about 17% plus the 8

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

bytes addition from convergence layer packet) causes concern for its efficiency. But as SMDS and 802.6 standards committees are turning towards more efficient AAL 5, AAL 3 /4 probably will go away.

AAL 5

Although designed on the basis of AAL 3 /4, it achieves three major goals. It reduces the byte overhead, minimizes the computer's cost in processing cells and it behaves like other data communications interfaces (as Ethernet and Fiber Distributed Data Interface FDDI) so that existing communications software can be easily ported to ATM. The SAR layer is a single bit in the ATM header which is 1 for end-of datagram and zero otherwise. The convergence layer has 8-byte long trailer as shown in fig 100.

Figure 100:AAL 5 Convergence format

The SAR layer has 1-bit in the header which is set for last cell in each packet. This one-bit is encoded in the Payload-type field of the ATM header. If the cell is a user cell, the last-bit in the payload type field is used to signal if the cell is the last cell of the packet or not. In the AAL 5 convergence layer, all the cells except the last cell of the packet are completely filled (all 48 bytes) with data. The last cell contains any data from the packet (if the packet's length was not 40+48n bytes long for some n) at the start of the cell, and eight bytes of convergence layer trailer at the end of the cell. Any bytes between the end of the packet and the trailer are filled with 0s. The trailer has four fields: a one-byte User-to-User Indication (UU) field, a one-byte Common part Indicator (CPI) field, a two-byte length field, and a four-byte CRC. The UU and CPI fields are currently unused and set to 0. The length field is the number of bytes of data in the packet. The CRC is a 32-bit code over the entire convergence layer packet including the pad and the trailer. The CRC is highly robust and can easily detect errors or lost and misordered cells.

ATM Signaling & Transmission

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ATM connection has to be set up between the source and destination before beginning any transmission which means a VCI/VPI pair must be assigned at each hop. This is done using the signaling protocol. CCITT recommended Q.93B as the standard setup protocol for ATM along the signaling protocol Q.931 for ISDN. ATM cells can not be transmitted over a data link, but some rules for how ATM cells are to be encoded and ATM cell boundaries marked are required. The cells may be encapsulated inside SONET frames. For example, to use the telephone network backbone, the cells will be encapsulated inside SONET frames or in T1 or T3 channels. In the encapsulation scheme each SONET frame contains a column of payload overhead (POH), followed by the ATM cells laid out in the rows of the payload. The payload column is pointed to by pointers in the SONET overhead. Because neither the row length nor the frame length is a multiple of 53, cells may straddle row and frame boundaries. The bytes in a cell must line up with the SONET byte boundaries. Individual cells are found by looking at the bytes to find patterns that match valid ATM header patterns; that is the bit pattern in first four bytes is legal and the fifth byte carries a valid checksum for the first four bytes.

Cell switches

Switches are the devices that transfer the data at the input port to the respective output port. Just because there may be many input ports and many output ports, the switches have to operate very fast. Cell blocking is a common problem in a switch where two cells from two input ports want to get to the same output port in the same time slot. To encounter such problems some switches are input-buffered and some are output-buffered. Output-buffered switches such as the Knockout and Gauss switches have better performance. Batcher-Banyan switches are input-buffered and one needs to be careful to make them perform well. Also a lot of the cost of switches are in the port controllers (which receive and transmit data) rather than in the switching fabrics. Optical cell switches use a WDM network and the transmissions are controlled through star couplers (meaning a coupler coupling each input Laser transmitter with all output receivers).

The cell switches and ATM networks using SONET frames are good for wide area networking. But in LAN or MAN environment, the networks are shared media networks (or multi-access networks). Just because the whole bandwidth is available to all nodes in the network, so the bandwidth needs to be

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

allocated and reserved through some arbitrating policy. There are many ring and bus topologies with various protocol designs for shared media networks that work at gigabit speed. Some examples of cell networks are Cambridge backbone ring (built as rings), Distributed queue dual bus (DQDB) (built as dual busses), Manhattan Street Network (fully connected mesh networks) etc.. Also there are LAN switches as AN2 switch which can be used to build a multi-hops local area cell network.

3.5 Multimedia applications

One important area of application of high-speed networking is *multi-media* applications. Such applications combine sound, video and computer graphics in an interactive environment. Examples are audio conferencing, video conferencing, virtual reality in which applications give the user an illusion of being somewhere else.

Sound

Transmitting sound over high-speed networks may not be the most interesting task, but still the bit rate varies with quality of sound desired. The telephone quality speech may need 16 Kb/s, but with wavelet compression methods this speed is going further down. Table 3-4 below lists the required bandwidths for transmitting various qualities of audio. Audioconferencing conversations need slightly higher bandwidth for the audio overtones which indicate that the user is in the room. Near-CD quality is almost like CD and CD quality is the best full CD quality.

Table 3-4: Bandwidths needed for different types of sound

Types of Audio	Frequency bandwidth
Telephone conversations	16 Kb/s
Audio-conferencing conversations	32 Kb/s
Near-CD quality Audio	64 Kb/s
CD- quality Audio	128 Kb/s

Normally, audio devices generate or play samples at fixed rates. For examples, digital telephone voice goes as 8-bit samples sampled at 8 kHz rate. CD audio uses sixteen-bit samples at rate of 44.1 kHz. One important need is to keep the delay of samples within a bound. The arrival of sample i+1 must

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

happen at the playing time of i+1 (after sample i) or within a bound (plus minus) of tolerable delay. Also another factor is the round-trip delay over the voice network. The round-trip delay when large will create confusion in conversation, but a value within 600 ms for telephone conversation is conveniently acceptable. Another issue is the echo timing. An echo returning within 100 ms of delay is acceptable, but anything greater than that is inconvenient. The difference between these two types of delays is very surprising. These limits are general , but still one audio sample may arrive later than the limit and the conversation will remain high quality.

Video

The communications of images, moving or still is an area of applications that takes high bandwidth. Video is normally transmitted in compressed form as frames at a rate of 30 frames a second or faster (to avoid flickering). All the data for a given frame must be sent into one sample, not many. But sometimes the information contents in a frame may exceed the bandwidth of the connection and that is when video encoders become useful. For example, fig 101 below shows in a video transmission frame 2 and frame 4 need more bandwidth (above the dashed line) and frame 1, 3 and 5 do not need such high bandwidth. The video encoders encode the video so that if one frame data exceed the bandwidth limit, enough information is sent to generate a partial image and then touch up using bandwidth of successive frames (that are below the bandwidth limit). So the encoder keeps track of two different images for a high frequency frame: the image the receiver has and the actual image. At present there are many schemes to support variable-bit rate video, MPEG being one example of vbr video encoding.

MPEG:

Moving Pictures Experts group (MPEG) is standards group working under ISO which specifies a standard way of transmitting compressed variable bit-rate video. The goal of the MPEG group is to develop a scheme for storing digital video (with its audio) which works well as a method of transmitting video over networks. At present MPEG schemes are defined only for low bandwidths (1.5 Mb/s), but higher bandwidth standards are coming. In short, MPEG uses three different frame formats: I, P and B frames. I (image) frames are complete bit images of a single frame and are occasionally sent to bound the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

errors in compression and transmission. The exact number of P and B frames that can be sent between I frames is configurable. P (predicted) frames update the image (using a predictive algorithm) from the last I or P frame. So given an I frame, a few successive P frames will try to track changes in that I frame. To further compress data for transmission, there are B (bi-directional) frames. B frames describe a frame in terms of the differences between the last and next I or P frames. It is simplest to think of the B frame as describing the difference from either the last high-quality image (generated from an I or P frame) or the next high-quality image depending on which image the current image differs least from. Note that because B frame can refer to either last or next frame, both the last and next P or I frames must be transmitted before the B frame can be transmitted. The ratio of B and P frames is configurable. A typical MPEG data stream will appear as a series of different frames bounded within a bandwidth limit. Loss of a P or B frame will affect the picture (because they are update frames) so much as loss of an I frame will. Also while encoding the data, outline information is more important than color or pixels well within an object. The less priority data can be sent using ATM cells with CLP set which is called layered encoding.

H.261 is another CCITT video encoding standard for transmitting motion pictures. Here the only difference from MPEG is instead of periodically sending I frame, image updates on a region of the screen is sent.

The high degree of compression of video images in standards like MPEG is achieved by permitting the image to be an imperfect copy of the original. Such compression is tuned to throw away pieces of the image which the human eye will not notice (lossy compression). But video encoders may take hundreds of milliseconds to compress an image and such delays may be unacceptably high in real-time video conferences. The qualities of the colored digital HDTV needs very high bandwidth (without compression 1 Gb/s and with lossless compression about 500 Mb/s).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

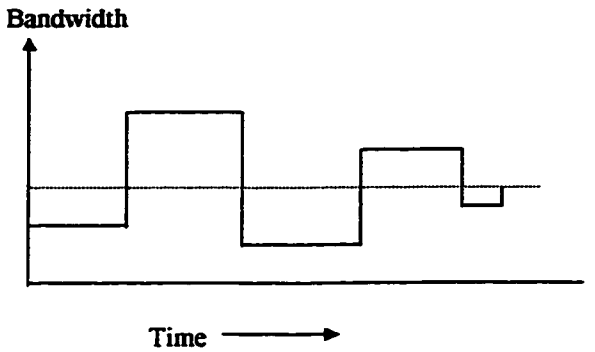


Figure 101:Bandwidth of video frames needing video encoding

Examples of multi-media applications vary from multi-media conferencing, large screen conferencing, interactive Graphics or virtual reality. To mention about virtual reality, NASA has developed software that uses the satellite photos of Martian surface and generate 3-D images of the planet. Using a headset apparatus which incorporates a 3D stereoscopic display, a geologist can get the sensation of Martian surface. Similarly a geologist can generate the images of earth's crust interactively. Such applications need to follow the following specifications:

1. Some applications may need a minimum standard bandwidth committed for the application to ensure a high-quality audio or hi-fidelity video image etc..
2. Some applications have requirements on maximum delay on delivery of data between the end points. Some applications may set it less than 400ms.
3. Continuous media applications have a timing requirement on the samples to be played back. If packets arrive earlier than the playing instance, it should be held in the buffer and the packet (in general) should be delivered before the sample's play time elapses. Of course the telephone networks maintain the same timing of the samples at every point in the circuit. But the requirement is not so stringent as to require a circuit switched connection. One example is Vat voice conferencing system on the Internet or simply stated, it can be used as a telephone over Internet. To avoid garbled voice, the voice packets are buffered and played every 100 ms. The time, samples are played is the playback time. Selecting the right playback time is a difficult problem. If the receiver chooses a playback point which is too early, then many voice samples will arrive too late and so the voice will sound

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

incomplete. If the playback point is too late, many samples will mix. Vat works by adjusting the playback point depending on the network delay. If the delay is large (i.e. many voice samples are coming in late), it increases the playback point, if it finds many voice samples appear before the playback time, it reduces the playback time.

3.6 Bottleneck in software for high-speed networking

The networking software needs to be efficient to be able to compete with sophisticated applications which demand higher speed. But simply stated, all networking software does is copy data from application to network layer and send it or vice versa. A major part of the networking cost lies in copying data between network interfaces, the host operating system and the network layers. Actually with time the data copies from memory to memory are becoming more expensive because processors are becoming much faster compared to the memories. Adding functions as super-scalar architecture or score-boarding does not help network applications much. Memory cache even does not help, because when a packet arrives, the interrupt code and the packet data (that need to be fetched from memory) are new (so, not to be found in cache). The networking code of BSD implements the networking protocols in an inefficient way. The system maintains the data in fixed-size memory buffers called *mbufs*. Each mbuf can store 112 bytes of data and 16 bytes of header thus making it 128 bytes chunks in total. Mbufs were used in implementing common protocol to make operation of adding and deleting easy. Mbufs can be chained into linked lists, and data within mbuf is obtained via offset and length counters in mbuf header. To delete data in some area of an mbuf, one has to erase the pointer and length. To delete entire mbufs, one needs to update the pointers in the link list. To add data, one has to populate an area in the mbuf and update the pointers in the header, but if no space is available then an mbuf has to allocated, the mbuf needs to be populated and the chain has to be adjusted to include in the linked list.

When an application sends data, it calls one of four variations of the *send* system call with a pointer to a buffer of data and the length of data. The BSD system copies the application data into a chain of mbufs and then passes this chain to the appropriate protocol stack for transmission. The code for

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

protocol stacks prepends additional header before passing the augmented chain to the network interface driver for transmission. The network interface driver copies the mbuf chain into local memory of the interface and transmit the packet. Receiving data is just reverse of this process. When an interface receives a packet, it signals its device driver to copy the new packet into an mbuf chain. The chain is passed to the protocol stack using an interrupt and the protocol headers are removed. The remaining chain is then placed I a queue where it waits to be read through a *recv* system call. At that time data is copied again to a buffer from mbuf chain. Thus at least two data copies are done for the transmission or the receiving process. Another disadvantage exists because the high-level routines copy the data into buffers before it knows how the lower layers need the data. If the lower layers need the data in a different format, another extra layer of data copy is normally inserted on an ad hoc basis thus adding another extra data copy operation. Also the OSI seven-layer approach was perceived by some Software people as the right model for building the networking protocols and thus it added more data copies. Also to implement a reliable version, BSD system was making a copy of the mbuf chain for retaining in case it needs to be retransmitted and this needed another data copy. To enhance efficiency, one needs to be able to copy the applications' buffers into interface in one shot. There are at least two ways: 1) memory mapped buffers on the interface, 2) copy-on-write memory management with DMA.

3.6.1 The Afterburner Interface

Suppose that one implemented a network interface so that it looked like some kernel buffer space i.e. processor's own memory then one copy would suffice for both inbound as well as outbound traffic. Jacobson analyzed the protocol implementation and said more costs were paid for interface control (e.g. start transmission, receive packet etc.) and memory copy. Afterburner design was for the interface trying to minimize these two costs.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

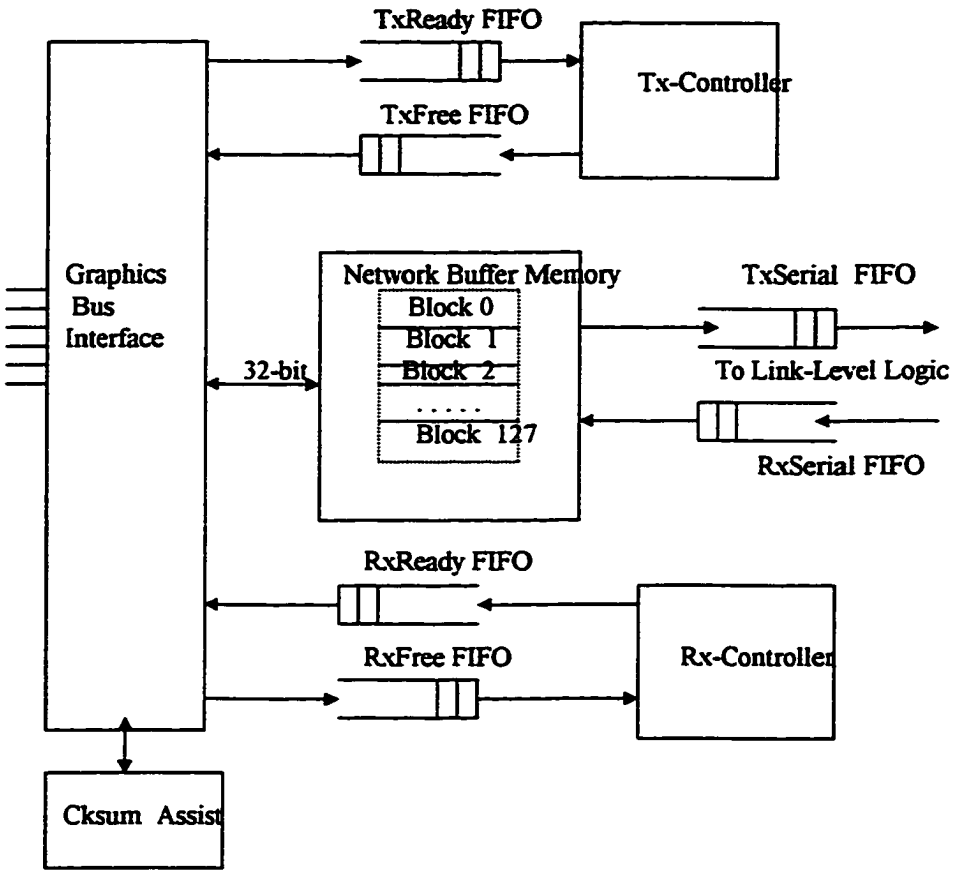


Figure 102:The Afterburner interface layout

The board is plugged into a memory bus with 32-bit data path (Actually the WITLESS card sits on the graphics bus on HP 9000 series 700 and maps its buffer into processor's I/O memory space, to avoid having to support error correction and interactions with memory cache). The layout of the interface card is shown in detail in fig 102. At the center of the interface is a triple-ported video memory, with two serial ports and one parallel port. This parallel port supports fast access to the graphics bus and the two serial paths provide transmitting and receiving paths to the link-level chipset on the right side of the interface (not shown).The link-level chipset contains the network-specific hardware to put data onto the network such as ATOMIC or ATM over SONET encoding hardware. The network buffer memory is organized as fixed-size 8-KB blocks. That makes the memory management simple and makes it possible to specify the buffer's length and address in a single word.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

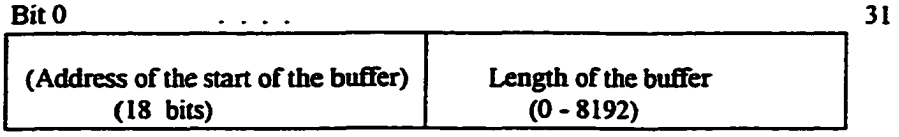


Figure 103:Afterburner FIFO Control Word Format

Before transmitting a packet using the Afterburner, the processor reads a word from the free transmission buffer (TxFree) FIFO, which contains the address of a free buffer. Once the processor gets the address of the free buffer, it populates the network buffer memory with the packet header, application data etc. and leaves a word (with the address of the buffer and length as in fig 103) in the transmission ready (TxReady) FIFO. The Transmission controller (Tx-controller) reads the word from TxReady FIFO and blows all the data on the network using the transmission serial port of the buffer memory. Once the data is sent, the buffer is cleaned by placing a word in the TxFree FIFO again. Now receiving data works in a similar way. Inbound packets are written into buffer and a token is written into receive ready (RxReady) FIFO. The processor is interrupted whenever the FIFO's state changes from empty to non-empty. When the processor has finished access to the buffer, it writes a word with the address and length of the buffer into the receive free (RxFree) FIFO. The control communications take place using single words on the FIFOs thus needing no locking mechanisms. Experiments proved that such interface cards increase the throughput to as high as 200 MHZ on a FDDI ring.

3.6.2 Copy-on-Write with DMA

The above Afterburner/WITLESS approach uses *programmed i/o* because the host makes the input/output copies into or from the network buffer and such i/o operations are interrupt-driven. In contrast to this approach, there is another design of i/o systems using the technique of *Direct Memory Access* (DMA). Some implementations minimize data copies using a virtual memory method called *copy-on-write* (COW) to manage the data buffers and a DMA network interface.

Copy-on-write methods permits sharing of pages in virtual memory inexpensively. If two or more applications want access to the same page simultaneously, each is given a pointer to the page and that

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

page in virtual memory is marked copy-on-write. When one of them tries to update the page, a copy is given to him where he writes his updates and thus protects all other users. COW can cut down the amount of memory copying for virtual memory.

DMA is data transfer method where a bus can transfer data from or to memory without disturbing the processor (just stealing some memory cycles). DMA schemes are typically optimized for moving bursts of data. The disadvantages are it bars the processor from memory accesses for some time and the i/o code to begin a transfer is quite complex. The principle of the COW method of communication is when an application requests a transmission the operating system marks those pages as copy-on-write and uses them as the operating system's data buffer. The operating system generates the header in its own memory and requests the interface to DMA to transmit the header and the buffer (from two different addresses). Assuming the application does not write anything into the buffer during this period (creating another copy), this scheme completes the transmission in one memory copy - memory to DMA interface. For the inbound packets, the operating system keeps the DMA interface informed of which pages are free. When an inbound packet is received, the DMA copies it into the free page and notifies the operating system. The operating system removes the packet headers and if the application's receiving buffer is page aligned, it simply changes the page table entry by replacing the user's buffer page address with the buffer address for the page referenced by the application. This scheme needs that the application uses buffers that are page aligned and are integral number of pages long. In some cases, language compilers and loaders can be modified to recognize data buffers and align them without changing the application.

There have been approaches to transmit or receive without making data copies altogether by combining page management principle of COW/DMA technique (no DMA required) and on-board memory of Afterburner/WITLESS approach such that the interface memory can be used as regular pages in the application memory space. The application would preallocate some of the interface pages as its sending buffer. Thus by populating the data buffers, the application has already placed the data in the interface buffer and hence no further copy is needed. For inbound case, no DMA is needed because the interface buffer is mapped as the receiving buffer in the application.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.6.3 Other Processor-memory Interactions

Although data-copies are the costliest operations, there are other interactions that also contribute to the cost, specially for small packets for which data copies cost minimal. These are:

1. Branch misses: Some processors operate using a pipeline of instructions where many instructions run parallel in the various stages of the pipeline. On a conditional branch, the next instruction to be executed becomes unknown and hence the pipeline stops implying a substantial penalty in the performance.
2. Context switches. When a processor switches from one task to another task, the old state needs to be saved, the new state has to be loaded, the code for the new task has to be loaded - as a whole it is a large overhead. The fewer the context-switches are, the better is the performance.
3. Interrupts. Interrupts occur when a device like a network interface signals a processor to indicate the device needs to be serviced.

So minimizing these above costly operations and thus minimizing cache misses is an important goal for good performance.

Another bottleneck on high-speed networking is real-time applications for multi-media systems which need time-bound response, but such guarantees are not offered by the networking protocols. To better understand the challenges in protocols for high-speed networking, one needs to see today's protocols. Among the various current protocols, TCP/IP protocol suite is most common for its two protocols Transmission Control Protocol and Internet Protocol.

3.6.4 Architecture of TCP/IP

The TCP/IP protocols suite was invented in 1970s is the internetworking protocol on the Internet, a global network connecting millions of networks across the globe. The basic architecture of the protocol suite is shown in fig 104 where Internet Protocol is at the center. IP is a datagram protocol used for internetworking. Internetworking is a way of interconnecting networks irrespective of the link layer or

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

MAC layer of the individual networks. An example of a small internet is given in fig 105 where two host machines can communicate in IP although the datagrams are carried by two dissimilar networks. Different networks as ATM, HIPPI, ATOMIC etc. may be required for different needs, but still they can communicate using IP (as long as they support IP). IP offers *best-effort* delivery of the datagrams without supporting error correction, guaranteed delivery etc.. Although currently IP addresses are 32 bits long, but the new IP version 5 in future will have 128 bits in address. The 32-bit long address is broken into three parts: a *network identifier*, a *subnet identifier* and a *host identifier*. Individual instances of a network technology are called *subnets* and a subnet identifier identifies the cable (e.g. an Ethernet segment) on which the host exists and the host identifier identifies the host uniquely within the subnet. The network identifier identifies a set of interconnected subnets uniquely. IP also supports packet broadcasting (sending to each host on the subnet) and multicasting (sending to a set of selected hosts) in the internetwork. IP needs that each network technology as Ethernet, HIPPI, FDDI etc. offer a standard way to encapsulate IP datagrams into the technolgy's framing protocol and also some address resolution protocol (ARP) that maps the IP address of the destination host to the local network address (e.g. an Ethernet address).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

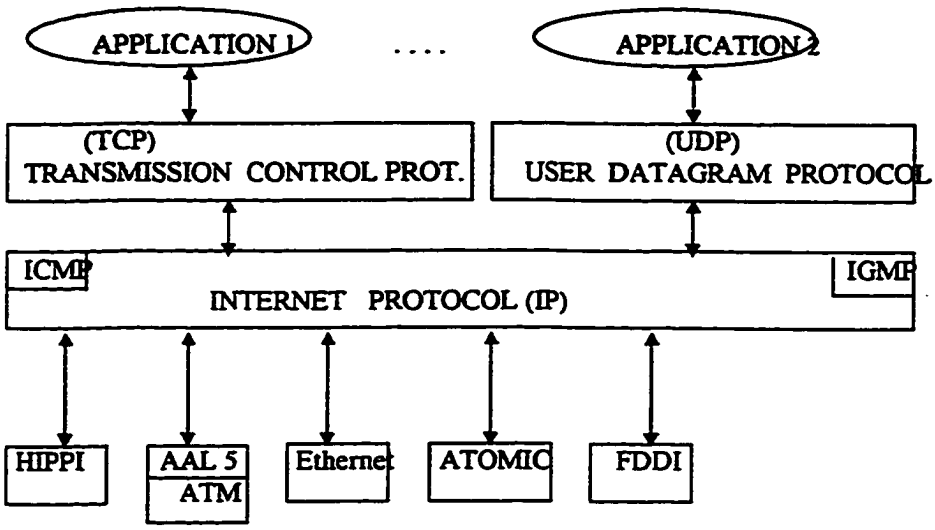


Figure 104:Architecture of TCP/IP protocol suite

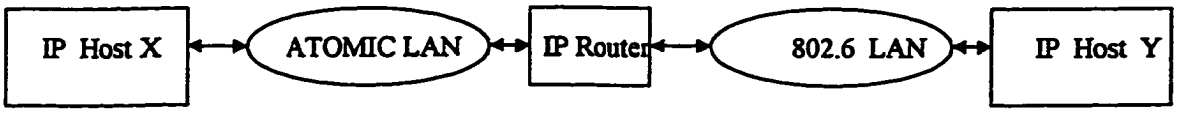


Figure 105:A mini Internet configuration

The rules for encapsulating IP datagrams in Ethernet frames is in RFC 894 and that for HIPPI frames is in RFC 1374 and for ATM (using AAL 5) is in RFC 1483. IP routers are used to interconnect subnets. IP routers will first pick up the datagram if it is meant for a host outside the subnet and then encapsulate it within a frame compatible with the second subnet and transmit it in the second subnet. Routing is done hierarchically based on the IP address. If a datagrams's destination is within the same network but on another subnet, then routing is done using the subnet id and host id. If the network is different, then the network id is also used in the hierarchy of routing. The routers update its routing tables by exchanging the network topology information every now and then using the special routing protocols.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In addition, IP also supports two more protocols, Internet Control Message Protocol (ICMP) which is used to carry error messages for IP transmissions between routers and hosts and the Internet Group Multicast Protocol (IGMP) which is used for multicasting. On top of the networking protocol of IP, there are two slightly different transport protocols called TCP and User Datagram Protocol (UDP).

UDP is a very simple connection-less protocol adding application addressing to IP as source ports and destination ports (16-bit identifiers). A port address and an IP address together uniquely identify an application . UDP is unreliable, error-prone and does not keep the data sequences in order. UDP offers a checksum, to check for errors in transmission, but use of that is optional.

TCP is much more complex, sophisticated and is connection-oriented. Like UDP, it also provides application addresses as a pair of port address (16-bits) and IP address. But it offers reliable, in-order delivery of the stream of bytes. TCP also relies on IP to transmit the data segments, but it expects to receive acknowledgments from the receiving TCP within a timeout period. If the sending TCP layer does not receive an ack within the timeout period, it will retransmit the data. The timeout period is estimated dynamically (an approximate round-trip time) by measuring the response-times for the prior acks. Also to ensure data error recovery, it adds checksums in the data segments. Applications as file transfer, remote logins etc. use TCP, but some unreliable delivery application, Multicast delivery use UDP (because there is no reliable multicasting).

The slow response of TCP/IP was investigated by Jacobson in details and he came up with methods to improve its protocol implementations. Some of the methods were better memory management (reducing data copies, cache misses etc.) and interrupt handling as discussed before, but there were also a few additional methods.

3.6.4.1 Reducing overhead of TCP/IP

Better lookup method

From the above discussion it is quite clear, that there are many informations TCP/IP will need to look up again and again from a table. For example, TCP must lookup to verify the connection block

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

whenever it receives a data segment. IP must find out a route for sending a datagram. In general, each lookup has a worst-case cost of $O(\log_k n)$, where n is the number of entries into the table and k is some base indicating the fraction of blocks that can be eliminated on average by comparison. These lookups constitute the major fraction of cost on memory access. There are two approaches to the solutions: 1) Use caches to hold up the tables wherever possible and thus avoid memory lookup , 2) Find better algorithms with even better running times. Studying the network traffic it is suggested that caches of just one control block or route is likely to achieve high hit rates. These caches are called *one-back caches* because they cache the entry found for one prior packet in the packet sequence. A one-back receiver cache keeps the control block that the last packet was destined for. A one-back sender cache keeps the control block from which the last packet was sent. Implementing these one-back receiving cache and one-back sending cache have improved performance well. As extension, keeping 20 last routes in the cache will boost the hit rate in the cache by probably 90%. For skewed traffic, a good lookup procedure will be hashing using open chaining, where the head of each hashed link keeps a cache of the last accessed block. In addition, fast-expanding routing tables are becoming a moderate problem - they are big and expensive to maintain. The new version of IP will keep the routing table small.

Reducing checksum computation

In addition, there is also another factor that adds to the cost: checksum computation., which is a memory drain requiring each byte of the packet be read and added into the sum. One step is to use the host's native word size and native byte order for the calculation. The checksum is calculated as a 16-bit one's complement addition of all the bytes. For example, if the bytes are

0x80, 0x81, 0x82, 0x83, 0x84, 0x85

when they are added as sixteen-bit words it becomes as

$0x8081 \bar{+} 0x8283 \bar{+} 0x8485 = 0x878A$

where $\bar{+}$ is one-complement addition. If the native byte order is opposite or the word size is different (say 32-bits), calculation can be done using its word-size and the result can be folded across sixteen-bits. Also the calculation of the checksum can be interleaved with data copy from interface memory to user buffer in such a way, that the adding can be done with the load and store operation in a

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

multi-instruction per cycle processor without costing any extra cycles. This method will reduce the cost of computing checksum and provide a boost to the performance only to the systems that execute two instructions per cycle. There have been other methods tried as dropping the checksum on LANs or using it at the trailer of data segment (the sender can continue send and still compute the checksum in parallel).

Prediction

People argue that because TCP has so many features as retransmission, window sizes, urgent data etc. it must be expensive to implement its protocol stack. Actually TCP's behavior is highly predictable and one can optimize the frequent path part of the TCP code in both sending and receiving implementation.

Fig 106 shows the fields in the TCP header. The source and destination ports are assigned at connection setup time and because TCP connections either always use options or no options, the data offset (Off) remains constant as do most of the control bits. In situations in which segments gets lost or reordered, the sequence number changes by the amount of data in the last segment received, and the window size typically does not change. The urgent pointer is only used if U bit is on, and it is usually not, and the PUSH bit (P) can be ignored if the receiver passes data upto the application promptly. Very few fields change unpredictably.

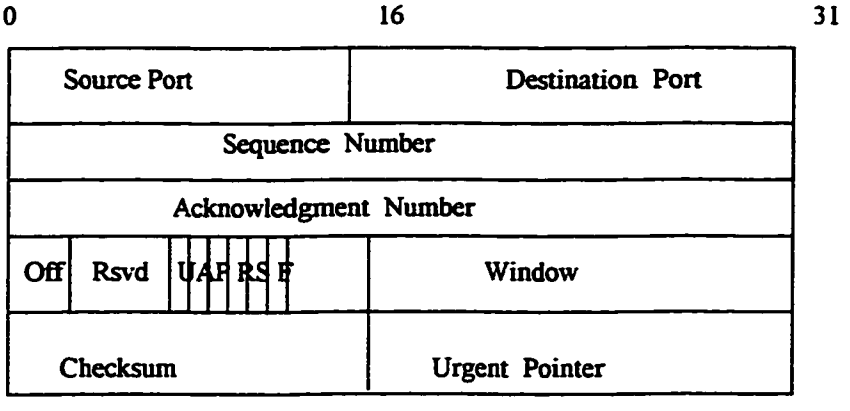


Figure 106: TCP fields in the Header

These observations led Jacobson to develop an algorithm for TCP receivers called *header prediction.* Header prediction looks for segments that fit the profile of the segment the receiver expects to receive next; namely segments that

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1. are for connections that have been established,
2. have only the acknowledgment bit (A) and optionally the push bit (P) set (Other bits on would indicate urgent data or that the connection was being closed),
3. are the expected next segment in the sequence (i.e., data in this segment starts where the last segment left off),
4. have not changed the window size,
5. are for connections that are not retransmitting data (if the connection is retransmitting, acknowledgment processing needs more effort),
6. are either acks for data or new data arriving, but not both (again processing both acks and data in one packet is more complex, and most connections are sending one-way at a time).

Once the control block for the connection has been located, these tests require five comparisons. A data packet that meets all conditions then needs only a few instructions (mostly to separate the header) and the data can be passed up to the application. Thus the extra cost of receiving a TCP segment, after the connection lookup and performing the checksum is tiny - a few instructions. Header prediction is an algorithm for the receiving TCP, but similar optimized predictive schemes also exist for sending in TCP. On the sending side, the application typically writes its data to some kind of a *connection handle*, a file descriptor or a socket. This connection handle can be mapped to a control block thus avoiding control block lookup. Because the destination is set, the control block can cache the IP route. Also, the outgoing segment can be predicted. The sending TCP can keep a template header, whose sequence number is incremented as segments are sent and whose acknowledgment number is updated as segments are received. Thus, sending means copying the template into the header, filling the checksum and populating the data.

Communication medium as optical fiber offers high delay-bandwidth product (easily of the order of megabytes), and so the limit on the window size of TCP had to be extended from 64 kilobytes. Currently an option has been defined to allow TCP's window size to be extended. Also the size for sequence numbers needs to be extended in TCP because at gigabits speed it takes 17 seconds to send 2^{31}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

bytes which may easily elapse on a connection and so the sequence Space has become too small. This problem is easy to fix and options have been set to extend this sequence space.

Currently, the cost of sending a datagram can be divided into a fixed part and a variable part. The fixed part is the header processing (handled by header prediction), the various table lookups (handled by caches and good lookup methods), plus time in the device driver and any interrupt costs and context switches incurred in transferring data from network interface to the application. Adding up these fixed costs results in about 150 instructions for both sending and receiving a TCP/IP datagram. The variable cost is in data handling, e.g. copying data, computing checksum etc. and varies from segment to segment. Putting all these together, the cost to process a TCP/IP datagram comes close to cost of copying data, provided the interrupts are not so expensive. A high speed processor should have no problem sending data at gigabits speed. But going fast is not enough today. Protocols need to be able to control the large amounts of data in flight and provide performance guarantees. The flow and congestion control mechanism in TCP is controlled by a slow-start algorithm. This probing algorithm requires the sender to keep a *congestion window*, which is the estimate of the traffic flow on the network. The congestion window is managed using a two-part algorithm. In the first part, the sender starts by sending one segment and waiting for the acknowledgment. When the ack comes back, the sender sends two segments, and waits on their acks, and then it sends four segments and waits on their acks. This exponential growth of the congestion window continues until the sender finds that a segment has been lost (from an overloaded queue). At this point, the sender starts the exponential pattern of sending from 1, but it grows only upto one-half the previous congestion window. After that the sender grows the window linearly, cautiously probing toward the bandwidth at which data was previously lost. This linear growth takes a long time. For this cautious probing, the congestion window setup algorithm is called the slow-start algorithm which is expensive for a gigabit link. On a high-speed link, some applications may need some kind of an open-loop congestion control scheme where a negotiation will take place at the time of opening the connection about the congestion window size instead of a feedback-based closed-loop control.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.6.4.2 Traffic shaping

It appears that a new model for the flow and congestion control is necessary for the new applications. The flow and congestion control model works in terms of flows. A flow is a delivery graph from a sender to one or more receivers. Packets forms the sender are carried down the graph through different routers to the receivers. Some of these packets are guaranteed packets and some are best-effort packets. Guaranteed packets must be delivered within time constraints with high reliability. Before sending any guaranteed packets over a flow, it is required to have some negotiation between the sender and the receiver and the internetwork. During this negotiation, the sender describes to the internetwork how it will *shape* the transmission of the guaranteed packets. Given this information, the internetwork will arrange for the individual networks (e.g. DQDB networks or HIPPI networks) and routers to schedule the flow's traffic so that the traffic has a guaranteed delay. Most versions of the model assume that best-effort packets can be transmitted at any time, without advance reservation. Such a model has three elements:

1. T*raffic shaping*: The sender has some way to describe the shape of transmission behavior of its guaranteed packets and to monitor its transmission behavior to comply with that shape. Given a description of a flow's traffic, the network can determine if the flow should be allowed or not.
2. *Special queuing schemes:* Routers have a scheme for managing their queues so they can make performance guarantees. Similarly, local subnets have methods that allow them to make guarantees.
3. A *setup mechanism:* There is a method to request guaranteed services from the internetwork.

The simplest traffic shaping schemes simply try to shape all traffic into isochronous flows, with regular amounts of data at regular time intervals. Some of the traffic shaping schemes are discussed below.

Simple leaky bucket

The leaky bucket scheme finds a way to shape bursty traffic into a regular stream of cells. Conceptually, a simple leaky bucket shaping scheme works as shown in fig 107.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

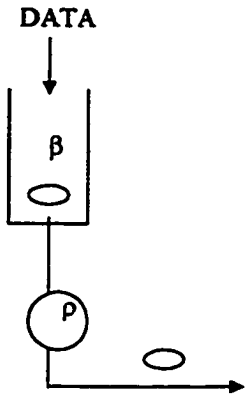


Figure 107:Model of leaky bucket

Each flow has its own leaky bucket. When data is to be sent over the flow, the sending host places the flow's cells into the bucket. Cells drain out the bottom of the bucket and are sent on the network at the rate of ρ cells per second. This rate is enforced by a regulator at the bottom of the bucket. The bucket size, β limits how much data may build up in the bucket waiting to catch the network. If data overflows the bucket, it gets discarded. The primary effect of such a bucket is to smooth a bursty flow into a steady stream of cells at regular time interval each cell being emitted $1/\rho$ units of time.

Simple leaky bucket was designed for cell traffic, but it can also be used to regulate datagram traffic. For datagram, one may think of the bucket size β and rate ρ being measured in bytes. When a flow sends a datagram of size x and there is space available in the bucket of size x, the datagram is placed in the bucket. When the datagram gets to the bottom, the datagram is held at the regulator for x/ρ time before it is sent on the network. The implementation of this model is very simple. The bucket can be implemented by a FIFO buffer and a timer will go on every $1/\rho$ time. At that time, if there is a cell in the FIFO, it will be sent out.

(r, T)-Smooth Traffic

The (r,T)-smooth algorithm was developed by Golestani as part of his stop-and-go queuing algorithms. In an (r, T)-smooth traffic system, a flow can inject upto r bits of data into the network in any T bit times. T is a fixed constant throughout the internetwork. The limit r, varies with the flow. If the next packet to be sent over the flow would cause the flow to transmit more than r bits in the current frame,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

then the current packet must held off for the next frame. A flow that follows the rule is (r, T)-smooth. The (r, T)-smooth traffic shaping is a slightly relaxed form of simple leaky bucket. Rather than sending a cell each $1/\rho$ times, the flow sends Tc/ρ bits of data each T bit times. It is a scheme which is very easy to implement.

A priority scheme can be easily added to the isochronous shaping scheme above. Such a scheme will help in controlling the errors for certain types of cells. Although isochronous shaping with priorities appears to be a useful service, some traffic shaping methods can support wider range of traffic than isochronous shaping. One of the example is token bucket.

Token Bucket

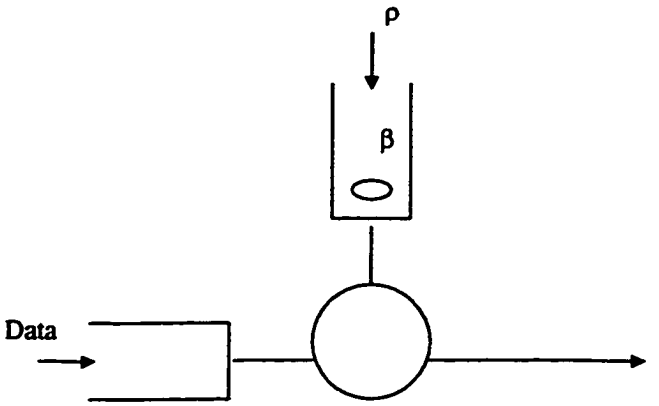


Figure 108:Token Bucket model

This model describes traffic patterns more accurately and so these schemes make the network run more efficiently. Fig 108 shows the details of the scheme which is a change on the leaky bucket scheme. The change is that instead of placing the flow's data in the bucket, the bucket is used to manage a device regulating the flow's data. Here, ρ is the rate at which tokens are placed in the bucket which has a capacity β. If the bucket fills, newly arriving tokens are discarded. When a packet is sent, it is placed in the buffer on the left. To transmit the packet, the regulator must remove a number of tokens equal to the packet size from the bucket. In cell networks, each token is equal to a cell. The effect of the token bucket

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

is very different on traffic shaping, to see which consider trying to send a packet of size b tokens ($b<\beta$) in the following cases.

1. The token bucket is full. The packet is sent and b tokens are removed from the bucket.
2. The token bucket is empty. The datagram must wait for b tokens to drip into the bucket and then it is sent.
3. The bucket is partially full with B tokens. If $b<= B$, then the packet is sent right away; otherwise it must wait for another b-B tokens before the datagram is sent.

From the example above, the difference between simple leaky bucket and token bucket is evident which is this model supports burstiness in the outgoing traffic, but bounds it (whereas the simple leaky bucket steadies a bursty traffic to a speed of ρ or less packets per second). This flow will never send more than $\beta + \tau/\rho$ tokens worth of data in an interval τ and that the long-term transmission rate will not exceed ρ. The other difference is that token bucket has no discard or priority policy. Token bucket discards tokens and leaves to the flow the problem of managing its transmission queue if the flow starts overdriving the regulator. Like simple leaky bucket, token bucket is easy to implement. The policing scheme has to be a little more sophisticated, because of the following. If policing is done by simply measuring the traffic over intervals of length τ, a flow can cheat by sending $\beta + \tau/\rho$ tokens of data in each interval. This is cheating because the flow would send $2\beta + 2\tau/\rho$ tokens of data in interval 2τ whereas it is supposed to send only

$\beta + 2\tau/\rho$ tokens of data during this interval. To place a further upper limit on the network flow, a leaky bucket may be placed on the output of the token bucket which will have a maximum transmission rate as C (a high limit). Then though the traffic will support burstiness, but will not exceed C and long term rate will be ρ tokens worth of data.

3.7 Performance guarantees

Among the traffic that pass a gigabit networks, there may be various types of requirements. Some packets may be time-critical, some may need bounded delay response etc.. Every traffic management scheme involves queue management in the devices connecting networks, possibly combined with controls

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(e.g. MAC layer support) within each network. It is useful to analyze the type of guarantee a queuing scheme may provide. Some schemes can provide *deterministic guarantee*; some offer *statistical guarantee*. Deterministic guarantees ask for a bound on the delay D between receiving and transmission. Another form of deterministic guarantee may be giving a minimum bandwidth (B_{min}) for the bit transfer rate or the worst-case loss rate. In contrast, *statistical guarantees* promise a certain percentage of data will be delivered within a specific delay D or that the average bandwidth available is ρ bits per second. Many applications can run on the basis of such statistical guarantees while permitting some exceptions. Also the statistical guarantee can promise far low delay because it does not need to worry about the worst-case delay, but deterministic guarantees needs to worry about the worst-case delay (or bandwidth) and require more complex queue management mechanisms.

3.7.1 Statistical Multiplexing

This new method of multiplexing depends on the strong law of large numbers. This law states that given a large set of uncorrelated flows, the total amount of bandwidth needed to satisfy all the flows is almost constant, even though the individual flows may vary meaning at any moment some individual ones will increase and some will reduce and this will eventually balance each other out with the remaining unbalanced part being very small compared to a gigabit per second (10^9 b/s). If the network knows the approximate statistics of the traffic it is currently handling (including guaranteed-performance traffic and non-guaranteed traffic) and its own capacity then it can determine if it can offer the requested guarantees (statistical) for another flow or not. In addition, interestingly it has been noticed that data traffic is not random in character but is some kind of bursty and self-similar and it tends to travel as a packet train. This pattern does not follow the law of large numbers and proposals of statistical multiplexing should take that into account.

Makrucki's Scheme

Makrucki has proposed a model for handling guaranteed flows in ATM networks. Although this is described in the area of ATM only, but it could be extended for packet networks also. Guaranteed sources are controlled by a leaky bucket with priority which generates a steady flow and never exceeds the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

specified flow rate at its output. If a source decides to exceed the leaky bucket rate, it must turn on the cell loss priority bit for cells in excess of its rate. If the source fails to turn on this bit, a traffic regulator in the network will detect the extra cells and turn the CLP bit on in enough of the flow's cells to remain within the guaranteed limit. Thus the cells with the CLP bit off are guaranteed to go through the network. Any unguaranteed traffic will have the CLP bit on.

A new guaranteed flow is allowed to start only if, for each link between the switches, the sum of the new flow's leaky bucket rate, plus the leaky bucket rates of the existing flows on that link, is less than some large fraction of the hop's bandwidth between two switches. As example, a link may allow new flows as long as all the sum of the flows on the link (plus the new flow rate) is within 50% of the link's bandwidth capacity. Queues in the switches have limits on the total number of cells from each particular VCI that can be in the queue at any time. The threshold is based on the number of cells that might appear in the queue if the cell were obeying its leaky bucket transmission rate. A If a cell with CLP bit on arrives when its VCI is over the threshold, the cell will be dropped. Cells with CLP bit off are always queued as long as there is space in the queue. Makrucki had simulated this scheme over a number of small network topologies using random Poisson sources mixed with bursty sources. The results suggest that the number of cells with the CLP bit set that are discarded can be kept low in most cases, though in problem cases as heavily loaded networks (85%) with large bursts of traffic and low queue thresholds high cell loss occurs (as high as 31%). This scheme is easy to implement (needs a counter per VCI and a FIFO queuing system). But it may make inefficient use of the network bandwidth because of its somewhat conservative loading requirements.

But, this simulation work assumes the traffic is mostly random rather than bursts and the simulation is done with a network of one to three switches. But experience shows that data traffic is more like a packet train and as an individual flow's traffic goes through more and more switches, it becomes compressed and bursty. Thus the simulations above may be misleading. Another simulation experiment of researchers have confirmed that TCP acknowledgments can be compressed together at the routers due to interference from other traffic. When such packed acks reach the source, it sends out again a packed string

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of packets which overload the routers. It is kind of clear that the same thing can happen in switches also. Cruz has shown that packet compression is generally a function of the number of hops in a network. Cruz studied the situation where traffic sources were shaped by token buckets with leaky bucket filters and packets were served by FIFO queues. He then proved that, in the worst case packet bunching occurs at each router and therefore bunches of packets from one router can interfere with other bunches at the next router. The longer the path, the worse are the interactions between the compressed bunches and the worse the potential queuing delay becomes. For a set of k identical sources with token bucket rate ρ and token buckets containing δ_0 tokens, the maximum per-connection delay is

$$\left[d\,(1 - k.\rho) + \delta_0 / \rho \right] \quad \left[\frac{1}{(1 - k.\rho)^h} - 1 \right]$$

where d is the max packet size and h is the number of hops in the path. Because ρ is the rate of flow (normalized with the bandwidth) and $k.\rho$ must be less than 1 if the network is not to be overloaded, the right term is a large number. The left term is close to 1, so the resulting delay is large and grows exponentially with the number of hops in the path. If data tends to get closer on one link, then how can the statistical multiplexing be possible? That is why Makrucki proposed leaky bucket shaping. Alternatively, one may use a traffic mixer which mixes traffic for different VCIs without changing the order of cells bound for one VCI. Making traffic more random makes statistical multiplexing more viable. Also another important factor needs to be mentioned with regard to the cell loss rate. The loss rate for an infinitely lived flow (i.e. computed over a very large period of time) will be quite different from the loss rate over a short interval of time. So the guarantee about loss rate needs to specify the interval over which the guarantee holds (e.g. the loss rate over a 10-mins period of time will not exceed 1 in 10^3).

3.7.2 Weighted Fair Queuing

Fair queuing is an idea developed by Nagle[34] who wanted to solve a problem in congestion control. Suppose there is a router attached to a link oversubscribed by unguaranteed traffic as TCP traffic. TCP implementations normally reduce the traffic as soon as they face congestion. But what happens if

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

there is one greedy TCP connection that takes advantage of the other well-behaved TCP connections and keep sending more and more data while others shrink their transmissions. Nagle wanted to find a way to manage the bandwidth of the oversubscribed link so that the greedy TCP connection will suffer and well-behaved TCP connections will still get good service.

The solution was to require to maintain a queue for each source host and to forward the packets from the head of the queues in a round-robin order. Empty queues are skipped, so given n active sources, the effect of the scheme is to allow each source to send one packet in each n packets sent over the link. The good thing about this feature is that if a TCP connection does not scale back its sending rate to 1, then only its own queue will overload. This original solution has some limitations. First, it ignored packet lengths, so a TCP that sent big packets will effectively get higher bandwidth (i.e. higher bits /sec) than a source of smaller packets. To remedy these problems, Demers, Keshav and Shenker [35] developed a revised version of fair queuing which comes close to a bit-by-bit round-robin service. Here, the n queues are served in round-robin order, where one bit from each queue that has a packet will be sent out. This scheme is much more fair than round-robin by packet and needs a packet to wait only n-1 bit times before starting to send it.

Given a particular bit-rate and a count of active connections, it is possible to compute when the last bit of the packet of length l will be sent (n.l). One limitation of fair queuing is that it gives each host the same fraction (1/n) of the bandwidth. Instead a weighted fair queuing will be better where the bandwidth may be divided into m -bit cycles, where m is greater than the number of queues (n), and then allocate the extra bits to those hosts that deserve extra bandwidth. Also as a last extension, the allocation of bandwidth should be by flows, not by hosts. This modified weighted fair queuing can be certainly a fair router and give strong performance guarantee to flows.

Parekh's formula

Parekh proved that, given a flow i, whose traffic was shaped by a token bucket with leaky bucket rate control and a network with weighted fair queuing in all the nodes, the queuing delay (D_i) suffered by data sent in flow i will have an upper bound of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$D_i = \frac{\beta_i}{g_i} + \frac{(h_i - 1)\, d_i}{g_i} + \sum_{m=1}^{h_i} (d\text{-} / r_m) \qquad (3\text{-}1)$$

where β_i is the flow's token bucket size, g_i is the weighted rate given to the flow (it must be greater than or equal to ρ_i, the speed at which the flow's token bucket fills), d_i is the maximum packet length for the flow, d- is the maximum packet length in the network, h_i is the total number of hops taken by the session, and r_m is the outbound bandwidth at hop m. Furthermore, this delay bound is valid, even if none of the rest of the traffic through the networks is shaped at all. Let us analyze all the terms of the delay. First consider a network in which the packet length is negligible. Assume that, all the traffic flows are initially quiet and have full token buckets. Then all the flows empty their buckets onto the network and continue sending at rate of ρ_i for flow i One can prove that this will maximize the queue length in the network. In this scenario, the queues will become longest in the network. Assume that for every flow the leaky bucket rate, C is infinite and $g_i = \rho_i$. Assume that the values of ρ_is all add up to the total bandwidth of each hop in the network. It should be clear that the burst of data represented by the flow's bucket size (β_i) will never drain out of the network as the network is the network is running at full capacity and data is flowing in at a rate of ρ_i. So the flow's delay in the token bucket is β_i / ρ_i (3-2). Now considering weighted flow, the delay is β_i / g_i (the first term in eqn 3-1). The next step is to account for the packet size. Under fair queuing a packet may be delayed from its appropriate transmission time by up to the maximum length of a packet (d-). The time that this delay takes is the length of the packet over the outbound bit rate (r_m) at the hop where the packet is being delayed, so the maximum possible delay through all the hops (h_i) taken by the flow is

$\sum_{m=1}^{h_i} (d\text{-} / r_m)$ (3-3) which is the third term in the eqn 3-1.

Also there are some packetization in the flow, which are bounded by the second term in eqn 3-1. The equation 3-1 is important for a few reasons. First, it says that using the most general shaping and admission scheme (token bucket with a leaky bucket regulator), strong guarantees on delay bounds can be given even if the rest of the network traffic is not shaped. Secondly, the queuing required in the network is

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

proportional to the worst-case delay. The queue length is approximately $g_i \cdot D_i$. This highlights the issue of serialization effects and the relationships between the packet sizes on a network and delay suffered by a session. As bit rate becomes large ($r_m \rightarrow \infty$), the last becomes negligible and the equation becomes only the first two terms in eqn 3-1 which means the flow's delay is made of the delay at its own token bucket and the delay due to its maximum packet size. Another important thing is possible this queuing delay bound plus the non-queuing transmission delay may be very close to the minimum deterministic guarantee that a network can give. Thus it is clear that coupled with weighted fair queuing the system can guarantee a flow a given share of the network bandwidth and a firm delay bound.

3.7.3 Jitter Control scheme

Weighted fair queuing can guarantee a upper bound for the delay but it does not give an estimate on the delay. Some queuing schemes can estimate the arrival time for a packet (sent at time t_0) to be $t_0 + x$ within a limit of jitter. Two different schemes to control jitter are presented here:

Stop-and-Go Queuing

Stop-and-go queuing was designed by Golestani and is designed to be used with traffic that has been shaped to be (r,T)-smooth.

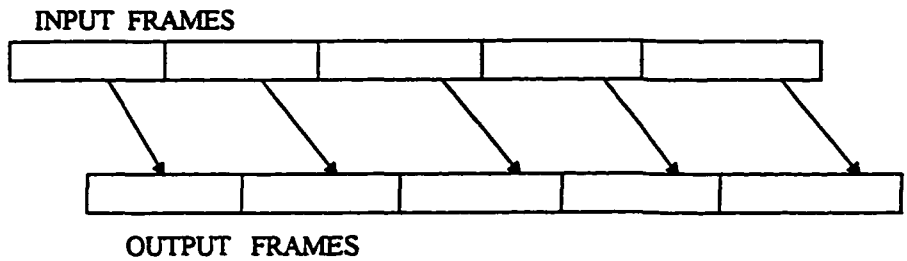


Figure 109:Stop-and-Go Queuing

The essential idea is to treat all traffic as frames of length T bits. A frame is considered to have arrived at a router only after the router has received the entire frame. A packet arriving in a frame cannot be forwarded until the entire frame is received. Stop-and-Go queuing requires that any packet in the incoming frame be forwarded in the first outgoing frame that starts transmission after the packet's frame has arrived. Figure 109 shows the frames forwarding policy. Consider the delay, D, experienced by a

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

packet. Define D to be a sum of three values: p, the propagation delay; Δ, a fixed delay; and δ, a small variance. Then observe that, under the queuing scheme, *if a flow is (r, T)-smooth in the arriving frame*, the traffic will be (r, T)-smooth in the flow's departing frame. From this observation, two points quickly follow.

First, the fixed delay, Δ, for a flow with a path of length h is bounded by the range

$$hT \leq \Delta \leq 2hT$$

because at best the packet is delayed at each hop by the time it takes to receive its frame, and at worst the incoming frame arrives just after an outgoing frame arrives has started. Because the interleaving of incoming and outgoing frames is fixed, Δ is constant. Note that the value of Δ is also the buffering required for the flow.

Second, the only source of variation in the packet's delay is the result of the packet being placed in a different place in an outbound frame than it occupied in an inbound frame. This means that δ must fall in the range $-T < \delta < T$. To verify, consider the case where the packet enters the network as the first packet in the frame. The greatest time displacement is achieved if, by the time it reaches destination, the packet is at the end of the frame, resulting in a displacement of +T. Similarly, a packet entering the network at the end of a frame can leave the network at the head of a frame, with a displacement of -T.

Because the propagation delay is fixed, these points show that a packet's delay in the network is strictly bounded by $p + \Delta \pm \delta$. If T is kept small, then the jitter is also kept small. However, there is a cost with bounded jitter. The jitter-bounded delay under Stop-and-Go is much higher than the average delay if Stop-and-Go were not used. For analyzing the reason, note that packets arriving in the start of an incoming frame must be held for the full time T before being forwarded. So all the packets that would arrive earlier would be delayed. In addition, Parekh has proved that the delay for Stop-and-Go queuing must be longer than that of weighted fair queuing unless Stop-and-Go queuing substantially under-utilizes the network. An outline of Parekh's proof is given here. One needs to view T as a time interval, not as bit-set. Note that the output from a token-bucket shaped output is (r,T)-smooth if

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$rT \geq \beta + \rho T$$

Solving for T, one finds

$$T \geq \frac{\beta}{r - \rho}$$

So the minimum delay through a Stop-and-Go network id hT. If one assumes the packet effects are negligible. then the worst-case delay through a weighted fair queuing network is defined by eqn 3-2. So if Stop-and-Go queuing is to have a better delay, then

$$hT \leq \frac{\beta}{\rho}$$

$$\text{or } h \frac{\beta}{r - \rho} \leq \frac{\beta}{\rho}$$

Solving for r, one finds that r must be greater than ρ (1+h). Given that ρ is, at most the average bandwidth of the flow, this implies that r is far greater than the average bandwidth (ρ) of the flow and therefore wasteful. Stop-and-Go and weighted fair queuing both need buffer space proportional to the length of the path. One important difference is that Stop-and-Go queuing requires buffering proportional to T, whereas at high speeds, weighted fair queuing needs buffering proportional to d-, so one would generally expect weighted fair queuing to need somewhat less buffering.

Jitter-EDD

Jitter-Earliest-Due-Date is a scheme proposed by the Tenet group at U.C. , Berkeley to limit delay jitter. It is similar in spirit to Stop-and-Go queuing but differs in some useful ways in mechanics.

When a flow is established, jitter-EDD gives each hop in the flow's path a queuing budget. b. Each hop computes its jitter bound j, which represents the maximum variation in the time a packet may take to get through the hop's service queues. At each hop, a packet is eligible to be transmitted b - j time after it arrives at the router and must be transmitted before the deadline time b. When the packet is

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

transmitted, the hop records in the packet's header the difference τ between the actual time transmitted t and deadline b. At the next hop, this difference is added to the hop deadline, so that the packet only becomes eligible for transmission at time $\tau + b - j$, and its deadline $\tau + b$. Thus the next hop makes up for the queuing jitter in the previous hop. As a result the flow's jitter at the receiver is decided fully by the variation in queuing delay at the last hop of the path.

Flows are regulated by a leaky-bucket-like admission scheme, so that packets are generated at rate ρ_i and thus are separated at admission to the network by interval $1/\rho_i$. The forwarding scheme described above will maintain this spacing through all the hops, because any packet that gets ahead of the $1/\rho_i$ spacing at one hop, due to the queue jitter, will be delayed at the next hop. Note that if a flow exceeds its agreed-on rate, it may overfill the queue and cause packets to miss their deadlines. To guard against this behavior and penalize flows that send packets faster than their agreed rate, ρ_i , jitter-EDD computes a packet's deadline as the maximum of $\tau + b$, and the deadline time of the last packet in the flow, plus $1/\rho_i$.

Jitter-EDD is mostly similar to Stop-and-Go queuing. The big difference is that whereas Stop-and-Go queuing work in terms of frames and delays serving packets until their entire frame has arrived, jitter-EDD computes a per-packet delay budget and delays serving the packet until the delay has elapsed. The queuing and delay bounds are similar.

In the beginning, Stop-and-Go queuing and jitter-EDD were both interesting because both of them offered deterministic delay guarantees to flows shaped with simple shaping schemes. But when weighted fair queuing with token-bucket shaping gives better delay and queuing bounds, these schemes are special for their ability to bound jitter. Proponents of jitter control method argue saying that jitter control minimizes the buffering needed to smooth delay at hosts and generally makes host implementation easier. Because the packets are supposed to arrive at the fixed playback time, so they need not be indefinitely buffered in the hosts. But others argue that memory is becoming cheaper each day and so the savings in the cost of the buffer is not significant. In addition, delivering data within strict jitter bounds does not significantly help host implementations, which must still worry about keeping data timing intact

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

because it must worry about the internal jitter in the local operating system, thus eliminating jitter in the network does not wipe out jitter in the host. As far as the memory issue goes, with days more hosts are being added to the data networks and memory prices are falling - so the saving may be significant due to the large number of the hosts, but may be insignificant due to cheap rates for memory. So this issue is still a question.

Weighted fair queuing, Stop-and-Go and jitter-EDD all suffer from one big problem: although they offer delay guarantees, the guarantees are often very long. An experiment done by Kurose simulating a four hop path through a T1 network carrying 48 identical 32 Kb/s flows showed that although well over 99.9% of all traffic going through a simple FIFO queuing network arrived within 15 ms, weighted fair queuing would have given a delay guarantee of about 65 ms and Stop-and-Go's delay guarantee would be more than 113 ms. Though this traffic pattern is likely to make statistical multiplexing using FIFO show good record, also it appears that the delay bounds offered by all the three schemes may be unacceptable to the applications, specially given that all the schemes need more overhead in routers than simple statistical multiplexing.

FIFO+

Clark, Shenker and Zhang have developed a queuing scheme called FIFO+ that tries to combine the delays of statistical multiplexing, with some delay guarantees of other schemes. They observed that both are work conserving, FIFO queuing and weighted fair queuing have the same average delay times through a network. But weighted fair queuing bounds the worst-case delay more tightly. The worse worst-case delay performance for FIFO comes from the chance that at each hop, a packet may be caught behind a burst of traffic from other flows. Over many hops, they get added up giving a bad worst-case delay. The concept with FIFO+ is to limit the accumulation of the worst-case delays on all the hops together. The approach is similar to that in jitter-EDD. Each hop groups flows into *classes*. Each class tracks its average queuing delay for the hop. For each packet, the hop calculates the difference between the average queuing delay for the packets of the same class and the queuing delay of the packet. This difference is added to the offset field in the packet header. Over the hops, this offset will record haw far ahead or behind the packet

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

is from the class's average. Each hop is understood to schedule a packet in its queues assuming as if it had arrived at the packet's real arrival time plus the packet's offset. As a result, packets which are ahead of the average, will be effectively pushed behind in the queue and ones falling behind will be pushed forward in the queue.

Simulations by Zhang suggest that FIFO+ queues give better worst-case delay bounds than FIFO over paths of more than one or two hops. In addition, these delay bounds are even lower than that for weighted fair queuing. It should be remembered that FIFO+ is a statistical multiplexing scheme because it does not give any isolation or guarantee. Applications that need strict delay guarantees will need to be satisfied with the longer delay bounds of weighted fair queuing.

So among the various schemes discussed above, Makrucki's scheme and FIFO+ give probabilistic guarantees. Deterministic schemes like weighted fair queuing, Stop-and-Go queuing and jitter-EDD give specific delay bounds. Weighted fair queuing bounds the maximum delay but does not tightly bound the inter-packet delay jitter. But Stop-and-Go and jitter-EDD offer ways to bound jitter though offering longer delays than weighted fair queuing.

Above the important aspects of performance guarantees were elaborated : the queuing algorithms inside the network or internetwork and the traffic shaping rules needed to make the queuing work. The next section expands on how a flow begins etc. in details.

3.7.4 Flow Setup and Routing in Internetwork

The important questions to answer before beginning a communication with a remote agent are:

1. *How does an application, or a host acting on an application's behalf, request a flow?* One needs to know some protocol and data structures which are required for interacting with the network for requesting a flow with application guarantees. This mechanism is interactive composed of requests from the host application and response from the network (yes or no etc.).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2. *How does the network figure out if to say yes or no for a flow request?* The network needs some source of information about current usage so that it can determine whether to accept or reject a requested flow. This is considered part of a bigger routing problem, in which the network determines if it can accept the flow, and if so, via what path or paths the flow should send its data.
3. *How to support multicast flows?* Multicasting is a popular demand for applications - for the ability of one source to send to many receivers. Flows will not be just between two points (source and receiver) but rather from one source to many receivers (hundreds or thousands even) who may join or drop out at any time. The flows will be tree shaped distribution graphs, where the root is the source, a few routers will be immediate children and receiving hosts and some routers will be grandchildren etc.. Before setting up any edge in the tree (which corresponds to setting up a flow to a node) guarantees are needed for the flow. Also the presence of multiple receivers obviously raises the heterogeneity issue. As an example, two receivers for a video broadcast, may like to receive at two different data rates. None of these problems is straightforward and multicasting makes them even harder.

At present, many proposals recommend various data structures and protocols to allow hosts request both unicast and multicast flows. Only incomplete solutions exist for routing flows and there is debate on how to handle multicast delivery of data.

3.7.4.1 Flow setup

Normally the beginning point of a flow setup is the host application. The host application requests a flow setup to the local Operating system which goes to the internetwork. At some point, the internetwork either accepts or rejects the request and some type of an identifier is returned which has to be used for any transfer of datagrams or cell transfer associated with the flow. For datagrams it is a flow-id and for cells it is VPI/VCI. The main purpose behind a request for a flow is to convey a description (called a *flow-spec*) of the flow's requirements into the internetwork.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.7.4.1.1 Flow request

There are some slight variations in the perception of what a flow-spec should be: it may be one or two values choosing from a set of standard behaviors or it may be a structure containing explicit values for each choice of requirements. These various types are basically the same, the main difference being how many different types of flows should there be. Some think that flows should be different only in a small number of ways. There can be video flows, voice flows, big file transfer flows and a few others and there can be some variations in their burstiness, delay requirements, bandwidths requirements, sensitivity to loss etc.. All switches and routers can be pre-configured with a dictionary of all these possible flow characteristics and when a host needs to ask it simply specifies the class and probably the bandwidth of video thus distinguishing between low-rate video or high-definition video flows.

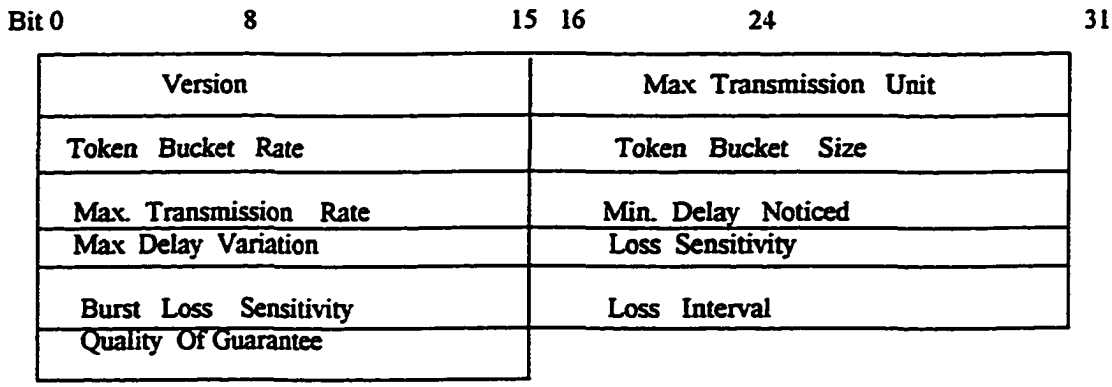


Figure 110:RFC 1363 Flow Spec structure

Some others add that such pre-built templates may not serve all the purposes because host applications may have application-specific requirements which may be hard to standardize on a global basis (because they are ever-evolving). For example, video-encoding may behave differently from one application to another. In this case it is advisable to let applications specify their own choices of the particulars. These two methods are similar to a point: both agree that basis needs bandwidth, delay and loss sensitivity need to be conveyed to the network principally whereas the other parameters may come from a pre-stated library or individually chosen. An experimental flow-spec has been specified by the researchers in RFC 1363 which is shown in fig 110. The version no specifies the revision number. This

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

flow-spec uses a token bucket with simple leaky bucket rate for traffic shaping. In the structure, one needs to specify the token bucket rate and the bucket size for the tokens which is kept in the front of the token bucket. This flow-spec does not give an option for maximum delay acceptable, but the host has to specify the minimum delay below which further improvement will not be noticed. The maximum delay is not specified because that changes according to the context of the application and the user. For example, a video conference between Tokyo and London may permit a delay of 750 ms, but a local video conference in the city of Tokyo will not permit the delay of 750 ms. So the maximum delay changes with context etc. The user can decide if the delay is acceptable or not by specifying delay variation. The minimum delay noticed limit is good to specify because that avoids having the internetwork search over-optimized paths. The flow spec specifies the flow's sensitivity to loss. Sensitivity is specified as sensitivity to total loss and sensitivity to loss of successive packets of the maximum transmission unit size during a particular loss interval. The loss interval is also required because long-term guarantees are not valid for short-term.

Finally, the flow-spec indicates whether the flow wants a statistical or deterministic guarantee. As has been stated before, the statistical guarantees are likely to be less than deterministic guarantees.

Once the network makes its request, eventually the network has to give its answer saying whether the flow-request was accepted or not etc.. There are differences in thought of what type of reply should the internetwork return. Some say it should a yes/no reply specifying if the request is accepted or rejected.

Others think the internetwork should always setup a flow that comes as close as possible to the host's specification. If this flow is not acceptable, the application can end the flow. Another intermediate approach is between this one and the first one where the internet refuses the request if it can not be supported without modification, but also it specifies the level of service that may be acceptable. A serious problem One problem with this approach is that it may be possible to trade off the multiple requirements of the flow in different ways (e.g. more bandwidth for more delay or higher loss for lower delay) and it will be hard to guess which one the host will prefer.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

There is another school which believes that the negotiations should become interactive. If the network can accept the flow, it does. But if it can not, it provides a feedback on which parameters are causing trouble and what possible values or options are acceptable. The user can make its choices and resubmit its request. However, there is always a possibility of dragging it on and on.

It may be worthwhile to keep it simple by only allowing yes or no responses to the flow requests.

3.7.4.1.2 Flow Establishment protocols

A number of flow setup protocols have been proposed. Some are internet-work setup protocol, some are proposals for ATM technologies etc.. A setup protocol must be able to support multiple receivers for a single flow with facilities as sharing a reserved bandwidth among many senders. In a voice conference, few people are typically speaking at a time, although there are many participants in the conference. Obviously one would like to have a setup scheme that allows all the senders to share a set of reserved bandwidth equal to say, capacity for three to five speakers. The setup protocol should setup the flows quickly, in tens or hundreds of milliseconds. Users expect the delays to be brief when joining a conference or changing their display from one video conference to another. The setup protocol should result in robust reservations. A setup protocol that results in reservations that are easily lost or ignored is not so useful. This means what be the reservation scheme it should have capacity to withstand temporary loss of connection
or a temporary crash of a router etc..

It may be useful to be able to change the properties of a flow once a flow is established. One good example is a video phone call where at one point a user decides she would like a better video quality and requests that the sending application send more bits of video. Suppose the old video rate was approx. b bits per second and the new requested video rate is B bits per second. At this point, one good way of handling the request is to continue the old flow and simultaneously open a new flow with higher transmission rate and if a new flow was successfully started then end the old flow (otherwise there will be

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

a long pause). If the available network bandwidth between the sender and the receiver is p, then p should be larger than B + b (the sum of two bandwidths). If p is less than B + b, then the network will fail to sustain two flows. In this case, it may be advisable to open the option of just raising the video bit rate (like a change flow-spec request).

It may be important to support advance reservations. Consider the case of a group of busy executives who schedule to meet on a multimedia conference for a given day and time. They would be willing to pay extra for such a reservation, but at this time network must make the bandwidth available for opening a flow.

A simple baseline proposal

It becomes easy to compare the merits of other various protocols if there is a baseline protocol to compare these with. Many research teams have used this particular strawman proposal though none wrote it up- probably because of some concerns for its limitations. Anyway, this simple proposal is a good model for setup protocol and nicely illustrates many issues.

This proposal suggests to enhance internet protocol like IP by adding a flow id field, and a flow spec option which can be added to the IP header. A sender who wants to set up a guaranteed flow, will choose a unique flow id, and starts sending regular IP datagrams with the flow-id field filled in. In the first datagram being sent, the user also has to mention a flow-spec for the flow in the flow-spec field.

The routers forward IP datagrams as before unless the datagrams have the flow id set. If the flow id is set, then routers forward the datagram based on the information the router has about the flow's requirements. Routers get information about the flow's requirement by reading the flow spec option in the first datagram. If a router receives a datagram with an unknown flow id, it forwards the datagram treating it as a regular IP datagram, but also sends a message to the sender asking for a copy of the flow spec. The sender retransmits the flow spec by adding the flow spec option to another IP datagram in the flow. An alternative proposal is for the sender to periodically send a datagram with flow spec option, say every n-th

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

packet. But in most cases all the routers that are involved will know about the flow. This scheme has been proposed for datagram networks, but it can also be used for cell networks. Keep in mind that most cell networks need both the delivery of flow information to each switch and the allocation of a VPI/VCI for each hop. As revised for cell networks, the sender places the destination address and flow spec in the first cell and chooses a VPI/VCI that is unique for the first hop in the network. Because VPIs and VCIs are unique per hop, and the sender always knows which VPI and VCIs are in use, it can always choose a unique VPI/VCI combination. Switches in the path read the first cell to find the destination address and flow spec, determine the outbound port for the connection, allocate a unique VPI/VCI for that hop and forward the cell on. Observe that, unlike routers in the datagram case, cell switches can not recover from crashes because the VPI/VCI typically contain no information about the cell's source, and so the switch can not request another copy of the cell from the source.

This approach has a few advantages. The procedure is easy to implement. There is no setup delay. The sender can simply send the setup datagram or cell and then start to send the data. Multicasting does not need much special handling. Datagrams or cells are simply forwarded down all the paths in the multicast packet distribution tree. If a flow is rejected down a particular path, one can either notify the source or silently suppress the flow. Recovery for router in the datagram case is easy. When the router comes back up, it can still forward the guaranteed traffic (using the destination address) while waiting for a new copy of the flow spec. While doing statistical multiplexing, the router can ignore the flow spec and forward data based on destination address.

However, this strawman setup scheme has many minor disadvantages. Sometimes the recovery scheme of the router does not work if there are many alternate paths to the same destination. If there are more than one paths to the same destination, some of which may be acceptable to the flow and some not. When the router comes back up after crash, it may use one of the unacceptable path for routing traffic. It may disrupt the path, and will surely hurt the quality of the flow. Also if flows are allowed to send data immediately after the setup datagram was sent, refusing a flow from a router may mean that the router has to discard all the packets that came in the mean time. This unduly overloads the router. Also it is not

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

clear how to handle heterogeneous receivers. If there is one black-and-white and one colored video receivers, how will the system separate the flow? Also there is no scheme for advanced reservations. There are two more mature protocols which are functionally more functional: ST-II and RSVP.

Stream Protocol - Version 2 (ST-II)

ST-II is the most sophisticated internetwork flow setup protocol that has been implemented already[36]. This is actually made of two protocols: a datagram forwarding protocol, called ST and a connection management protocol called ST Control Message Protocol (SCMP). ST just forwards datagrams down the routing trees created by SCMP. The following discussion is on SCMP.

There are seventeen different SCMP messages, corresponding to different flow management services that SCMP provides. These messages can be used to set up a flow to one or many receivers, remove one or many receivers from a running flow, add one more receiver to a running flow, change an existing flow-spec, and manage a flow by doing error recovery and status checks. In ST-II, a flow's distribution graph is source specific - only an individual sender may use a particular distribution tree. If more than one sender is transmitting to the set of receivers, so many flows and distribution trees have to be setup first.

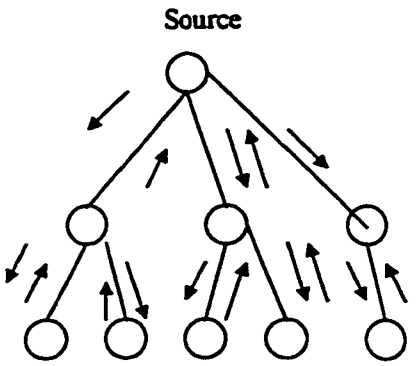


Figure 111:Sequence of flow setup between the hops (ST-II)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Flow setup is done on a hop-by-hop basis. Each router in the path must negotiate with the routers downstream from it in the flow distribution graph to establish the flow. Most flow setup operations work in two stages: a request from the sender works its way hop by hop down the distribution tree, and then responses from the receivers work their way back up the tree. Fig 111 illustrates the sequence of flow setup calls over the network from the sender to the routers down the distribution tree.

For understanding how ST-II works, consider how ST-II sets up a flow. A sender starts the process by sending a CONNECT message, which contains a flow spec, a list of all the flow's recipients, a flow name (a unique flow id) and a proposed hop-id (a hop-specific shorthand identifier for the flow) to the router on the sender's network. [3] The router immediately acknowledges receipt of the CONNECT message. The router also decides if the proposed hop id is acceptable and if so, it accepts the hop id. If the hop-id is not acceptable, it starts a negotiation with the sender to choose a better hop id. Concurrent with negotiating the hop-id, the router examines the list of destinations. If there is a destination the router can not reach, it sends a REFUSE message for that destination. For those destinations it can reach, the router sorts the destination list by the next hop routers through which each destination is reached. For each next hop, the router determines if it can meet the requirements of the flow spec over that hop; if not, the router reduces the requirements of the flow spec to match what is available. The router then sends its own CONNECT message to each of the next hop routers. Each CONNECT message contains the revised flow spec for the next hop and the destinations list reachable through that hop router. These grandchildren routers again in turn examine the CONNECT message and decide whether to accept the proposed hop id , and which destinations can be reached etc..

Eventually, these CONNECT messages reach all the way upto the destinations. Each destination checks the version of the flow spec that it receives and either accepts or refuses the connection, using an ACCEPT or REFUSE message. As part of the acceptance process, the destination can reduce the requirements flow spec that was received to match its own requirements (e.g. black-and-white video etc.).

---

[3] Here it is assumed that the sender has only one router as child in the distribution tree. If there are more, it should send one connection message to each router.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The ACCEPT or REFUSE message works its way back hop by hop to the sender. At each hop, the router examines the message to see which destination has accepted or rejected the flow and if the flow specs have been modified. If the destination has accepted it, the flow has been accepted. If none of the destinations through a router accepts, then the partially established flow is turned down.

SCMP is hop by hop reliable. All messages are retransmitted if the next hop does not acknowledge them. Furthermore, once a flow is established, routers participating in the flow keep track of each other, and they can dynamically reroute flows when routers fail and reload information about flows into the routers that reboot. In spite of its complexity, ST-II has been implemented by many research teams. Their experiences suggest that although implementable, its complexity is big enough to make it unlikely for implementation on internetwork. One concern is that ST-II has a very large state space, in which a flow could be simultaneously adding new receivers, removing existing receivers and updating flow-spec which makes coding very complex. One problem is that ST-II does not scale well to large multicast groups, because each destination must be explicitly listed in the CONNECT message. As a result, a connection to a few hundred hosts requires the sender to process hundreds of ACCEPT or REFUSE messages. Also all messages to change streams must originate at the sender, which means the sender must periodically be interrupted from sending to process messages from receivers asking for changes in their flow.

ST-II was designed for datagram internetwork, but works well with cell networks also. ST-II needs that a flow be established before one sends and that senders request flows to receivers. One has to build an ST-II delivery tree using hop-by-hop cell connections between hops.

As a whole, probably ST-II is too sophisticated protocol. Excepting advanced reservations, it offers most of the expected services. Receivers can modify flow specs, multiple receivers are supported, flows are robust. There is some question how fast flow setup is. But its design places extra burden on the sender to maintain the flow and forces each sender to establish a different flow.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RSVP

The resource ReSerVation Protocol (RSVP) is another implementation of the solution for setup problem. Instead of having the sender manage the flow, RSVP has each receiver manage its part of the flow's distribution graph. RSVP was developed at XEROX and the University of Southern California [37]. In RSVP, senders easily multicast their data flow and the internetwork decides which receiver should receive the data. Along with sending data, the senders periodically transmit *path* messages which include a flow spec describing their flows. RSVP assumes that the flow may be multi-layer encoded (like video) and there may be one flow spec for each layer of encoding thus simultaneously many flow specs for the same flow.

If a receiver is interested in receiving many senders' flows, a receiver tunes to the proper multicast group and looks for the *path* messages. Based on the path messages, the receiver decides what part of the senders' flows it would like to receive (e.g., just black-and-white bits or full color) and generates a reservation message, which has a filter and a flow spec for each sender's flow. The filter indicates what parts of the individual sender's flows the receiver wants to receive, and the flow spec describes the properties of the path that each filtered flow will require. The reservation message is propagated I the reverse direction along the route from the sender, through the tree established by the path messages. There are three classes of filters. In *no-filter* mode, the sender's flow is not filtered at all - all datagrams from any sender are forwarded to the receiver. In *fixed filter* mode, a sender's traffic is filtered according to a fixed filter for the duration of the receiver's reservation. In *dynamic-filter* mode, the receiver can change how a sender's traffic is filtered over time.

Filters have three important purposes. First, they provide support for heterogeneity. Receivers at the end of slow links can still participate in flows by using a filter to restrict what portion of a flow is passed to it. Second, dynamic filtering allows receivers to modify flow properties. A good example of this is where the receiver alternately listens to many flows: it can change filters to "drop all packets from sender A" to "accept all packets from sender B" to change which flow it is listening on. Thirdly, filters are used to try to reduce load and improve bandwidth management.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

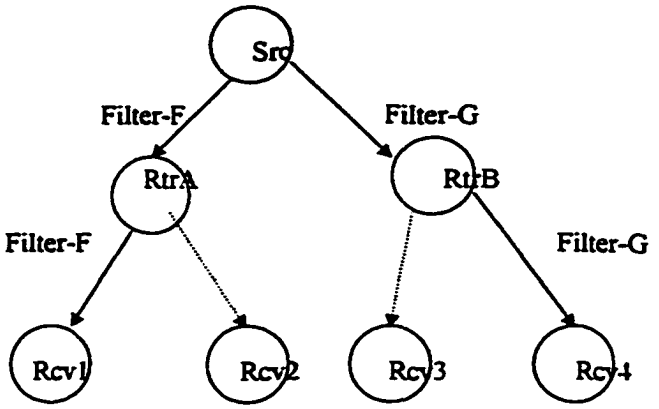


Figure 112:RSVP delivery tree from Src to Rcv1, Rcv4

To see how the filters are used, one has to look at how reservation messages are handled. When a receiver generates a reservation message, the message is forwarded through the internetwork towards the senders. Each router on the path checks into the reservation message. If the reservation request has no filtering or fixed filtering, the router tries to aggregate the reservation with other reservations with the same filter (or no filter). If a reservation already exists with the same filter for the same sender, the new receiver is simply added to the flow distribution at the current router.

Otherwise, the reservation needs to be forwarded farther up the flow distribution tree. Fig 112 and fig 113 illustrate how this aggregation of reservations is done. Figure 112 shows an existing routing tree through two routers (RtrA and RtrB) to two receivers Rcv1 and Rcv4, who have reserved particular filters. In figure 113, a new receiver Rcv2 has been added which asked for filter F (same as the one Rcv1 is on). So the reservation message from Rcv2 needed to go no further than RtrA. Comparing RSVP with ST-II, one can see that a sender in RSVP is involved in much less activities for managing the delivery tree.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

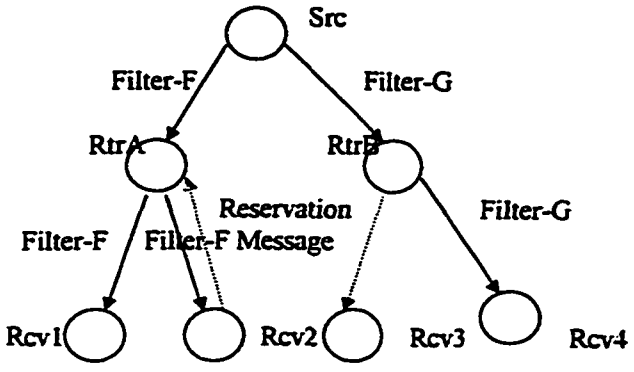


Figure 113:Adding a reservation from Rcv2 using RSVP

RSVP uses *soft state* to maintain information about reservations. Soft state is information that is periodically refreshed by interested parties. In RSVP, senders and receivers refresh the state at routers. Senders are needed to periodically retransmit path messages to allow new receivers to learn of the flow, to remind routers that the flow exists and to adapt to routing changes. Receivers periodically retransmit their reservation messages to remind routers of their reservation. An advantage of soft state is that it makes error recovery easy. Receivers or senders that crash stop reservation and path messages. So if a reservation is not refreshed within a given period of time, the router discards the reservation. Also when a router crashes, its state will be refreshed by the periodic path and reservation messages after it reboots.

RSVP does not fit easily into cell networking model as ST-II. RSVP expects receivers to be able to listen to flows for deciding if they want to reserve resources. But in cell networks, the senders will establish the connections to receivers. RSVP comes somewhere in between ST-II and the baseline strawman proposal in complexity. It offers same functions as offered by ST-II, but like ST-II lacks support for advance reservations. But RSVP allows multiple senders to share the same bandwidth and it places a much smaller burden on the senders.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.7.5 ATM Connection Setup

Q.93B is a simplified version of the ISDN call setup protocol, Q.931. ATM forum and CCITT jointly created the revision of this specification. In 1994. Q.93B supports dynamic creation of ATM virtual channels. If a system needs a virtual circuit to be created to another system, it uses Q.93B to establish the virtual circuit. Once created, the virtual circuit stays open until either it is closed by its users or a network failure occurs. These dynamic virtual circuits (also called *demand virtual circuits*) can be contrasted with *permanent virtual circuits*, which are reestablished after network failures. Q.93B supports the creation of three types of connections. A requester can establish a unidirectional channel to a destination, a two-way channel to a destination (only if the bandwidths in both the directions are equal), or a unidirectional multicast delivery tree. Q.93B is set of rules for how the sending and receiving ends of a virtual circuit negotiate connectivity with the ATM network. How switches within the ATM network communicate with each other is not yet standardized. They may use Q.93b or their own protocols, Q.93B defines the services the setup protocols must offer. The exchange of Q.93B messages required to establish a simple one-way connection between a sender and a receiver is shown in fig 114.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

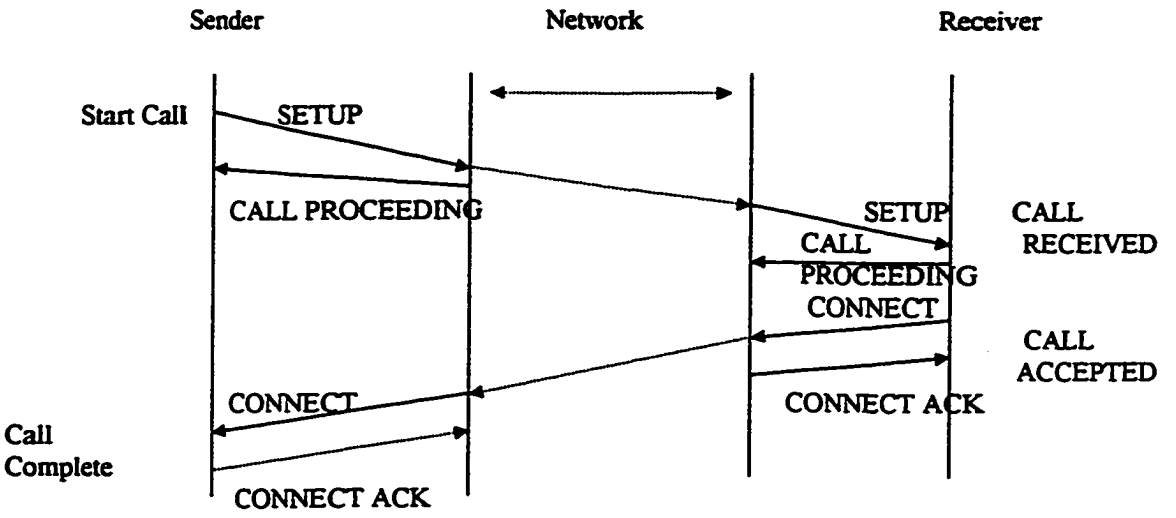


Figure 114:Q.93B Connection Setup

The sender begins by sending a SETUP message. The SETUP message contains information about the call, including a flow spec, which AAL is to be used (and AAL-specific parameters like the largest packet that can be sent if using AAL 5), and the address of the receiver. The network acknowledges the SETUP message with a CALL PROCEEDING message. The primary purpose of this message is to acknowledge the SETUP message, but it also contains the VCI/VPI for the connection.

Although the flow spec is capable of expressing traffic behavior in terms similar to that of token bucket shaping, the current standard requires that the cell transmission rate be expressed as a fixed rate (like in simple leaky bucket) and the other fields be ignored. The destination address can be expressed in a number of forms, including E.164 addresses (which is like an extended telephone number) and IEEE 802 addresses.

The network notifies the receiver that a call is requested by sending a SETUP message to the receiver. The receiver acknowledges with a CALL PROCEEDING message and then decides, based on information in the SETUP message, whether to accept the call. If the receiver decides to accept the call, it sends a CONNECT message to the network. The network acknowledges the receiver's message with a CONNECT ACKNOWLEDGE and in turn sends a CONNECT to the sender indicating the call is

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

established. The sender may optionally acknowledge the CONNECT message with an ACK. If there are errors in the negotiation, the typical response will be RELEASE COMPLETE from the network meaning the call has failed and all information on the call should be discarded.

To ensure reliable transmission of messages between the endpoints and the network, the SETUP and CONNECT messages are retransmitted if they are not acknowledged within a sufficient period of time interval. The sender can not begin sending data until the CONNECT message has been received, so if the network detects that the sender id sending data on the new VCI, that is taken to be an implicit acknowledgment. With all the length protocol, people expect it may take sufficient time (almost a second) to setup a call fully.

Multicast calls are achieved by establishing a one-way connection and then adding further receivers using the ADD PARTY message. Multiple new receivers can be added in parallel. All receivers receive the same type of service requested in the original flow spec. Two-way connections are accomplished by including a flow spec for the reverse channel in the initial SETUP message. As a setup protocol, Q.93B is somewhat minimalist. It supports the essential services. Flow requirements are specified and multicasting is supported. But the setup times are long and the fixed bandwidth may be wasteful. Also it supports only homogeneous receivers. It is not clear if these limitations can be worked around or not. Possibly by caching the connections, the long setup times can be worked around. When a connection is closed, the connection may be left open in the cache for some further period of time before it is really closed so that another application can readily use that. This technique has been very useful in trying to implement IP over X.25 networks. Of course, in tight situations leaving an idle connection open can be expensive in terms of unused bandwidth and idle information in the switches. Even if two applications want to talk to the same receiver successively, it is not known if their requirements will be identical. A low-bandwidth connection after a remote login application may be of little use for a high-bandwidth file transfer application. The wasteful fixed-rate allocation of bandwidth can be improved using priorities. However slow receivers in a multicast group may still be overwhelmed by unwanted bandwidth.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.7.6 Routing

ST-II, RSVP, Q.93B all assume the existence of a routing system that will guide the traffic through a path in the network. But building such routing systems is very difficult. Routing has been the problem in the past of determining if a path exists between two points in a network or not. Supporting flows changes routing into a far more difficult problem of determining if a path satisfies a flow's requirements. Unfortunately, routing on multiple requirements can become an NP-complete problem in the number of links. Therefore the current challenge in routing is to find ways to route flows in an inexpensive way.

Problem

Considered abstractly, routing is the problem of finding a path through a graph, subject to one or more constraints where the graph is an abstraction of the network's connectivity. Today's protocols like The Bellman-Ford algorithm (implemented in protocols as the Routing Information Protocol (RIP)) and Dijkstra's algorithm (implemented in protocols as Open Shortest Path First (OSPF) and intermediate-system to intermediate-system (IS-IS) routing protocols), are capable of routing based on one constraint. In other words, given a connectivity graph a single metric associated with each link in the graph, both Bellman-Ford and Dijkstra's algorithms can quickly find a path between two points that minimizes the sum of link metrics in the path. For example, given the simple network shown in fig 115, both algorithms will discover that two paths exist from A to B and choose the path that minimizes the sum of the metrics. Thus if $\gamma_1+\gamma_3 < \gamma_2+\gamma_4$, the algorithms will pick the route (A, R1, R2, R3,B); otherwise they will pick the route (A,R1,R4,R3,B).

The major difference is that Dijkstra's algorithm distributes complete routing information to all routing agents, whereas the Bellman-Ford algorithm tries to minimize routing information by requiring routers to pass along information only about the best routes that they are aware of. Both approaches have strengths and weaknesses. Having full topology information is necessary to do routing for flows so it may be that Dijkstra's algorithm will be preferred for routing flows.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

It is important to note in any of the two algorithms the metrics can be anything (should be a cost factor that one is trying to minimize). It can be measure delay, bandwidth, cost of the link or any other type of nonnegative integer metrics. Routing for flows complicates the problem further in two ways. First, each link has to be described in terms of multiple metrics. Second, whether a link can participate in a path depends on the requirements of the flow being routed.

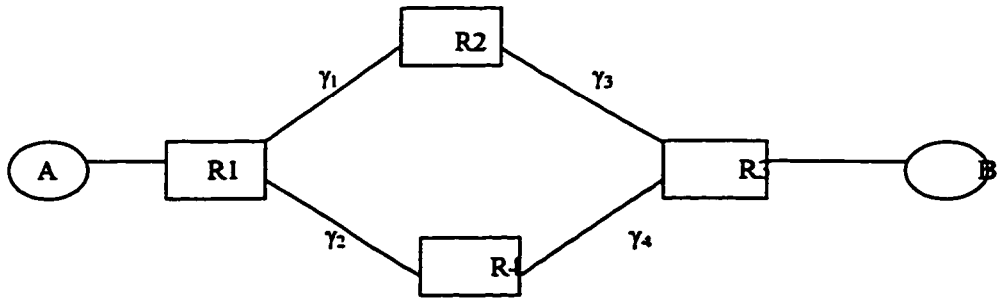


Figure 115:A Simple network (of routers and hosts) with link capacities

To illustrate why these changes are a problem, consider the following situation: Fig 116 illustrates the same simple network, but with each link having three metrics: a delay δ, a delay variance v, and the available bandwidth, β. Such a multi-metric description of a link is sometimes called a *hop spec.*

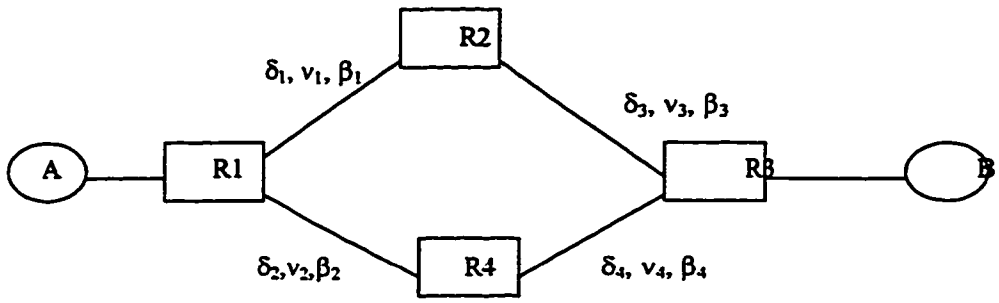


Figure 116:Simple network with constraints

Now suppose that we are trying to find a path for a flow, f, from A to B, which has the requirements that delay be minimized, delay variance be less than v_f and a minimum bandwidth of at least β_f. Because both

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Bellman-ford and dijkstra's algorithms can minimize only one metric, like a path to minimize delay. But it may happen so that this path has very low bandwidth than required or high delay variance. Similarly, if we routed solely on bandwidth or delay variance, we might pick a path with an unacceptable delay.

What is really needed is an algorithm which, given a flow-spec and information about the links in the network, chooses a path meeting the requirements of the flow-spec. In this simple example, one such algorithm will be to compute the combined metrics of all the paths and compare the combined metrics with the flow-spec. This algorithm works for very small networks, however. An efficient algorithm for large networks is still needed.

Solutions for large networks

A few proposals have been made on finding an algorithm for routing flow in large networks, but all have some limitations.

One proposal is to convert the multiple metrics into one metric. Given the flow spec and a complete map of the network, one can review the capacities of each hop and based on the flow's needs generate a single metric for each hop. Then one can run Dijkstra's algorithm on the network using the flow-specific metric. This strategy is a version of *policy routing*. Policy routing finds a path based on Boolean constraints which may differ on a per-flow basis. Because all the constraints are Boolean, one can examine each hop and generate a cost metric of 1 if the hop meets the constraints and infinity if the hop fails to meet the constraints. (Recall that the routing algorithms find the shortest path, so an invalid path should have an infinite length.) If a valid path exists, Dijkstra's algorithm will find it. Generally speaking, it may not be possible to convert all metrics into a single metric for all flow's requirements. However such hops can certainly be eliminated from consideration in this fashion. For example, if a flow's required bandwidth is β, one can certainly eliminate all hops from the network which has capacity less than β. Also it is possible to build networks where certain constraints can always be met. The world's telephone networks, for example, are carefully engineered so that the paths that a phone call may take will meet the delay requirements for voice. It might be possible to construct an internetwork in such a way that

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

meeting a flow's requirements is a matter of solving for a path using just one metric. If one can not reduce the problem to one of solving for one metric, maybe it can be reduced to one of two metrics. Castineyra has developed an algorithm that can find the set of paths that minimize two metrics on moderately complex topologies in less than a second. In this context, *minimizing* has a special meaning. A minimum point is a point a, b where any point with a better value than a must have a worse value than b and vice versa. So Castineyra's algorithm generates a list of potential minimum-cost paths. How one chooses a particular path from this list is not yet determined.

Multicasting and Multi-objective Routing

If finding a path for a flow between two points is hard, finding a delivery tree from a source to multiple destinations which meets the divergent requirements of the different destinations is even harder. Q.93B supports delivery to multiple destinations but totally ignores the problem of differing receiver requirements.

If one examines ST-II carefully, ST-II's approach is that each hop computes the point-to-point path and then combines all destinations that share the same next hop. Note that ST-II does not know each receiver's requirements until the receiver gets the CONNECT message and can modify the flow spec, so the path ST-II chooses for establishing the path may not be optimal. Hence ST-II supports differing receiver requirements, but may over-allocate resources in the network. Because it is receiver-based, RSVP will typically find the best path between the existing delivery tree and the receiver. And because the receiver initiates the reservation, the path will reflect the receiver's particular requirements. Probably this approach of RSVP will make better use of the network than ST-II, but it is not known so well.

3.8 Cellular and Cordless Communications

In last 15 years, the number of cellular subscribers has risen from 0 to 80 million all over the world. In some countries more than 25 percent people use mobile communications. The cellular and cordless phones, both new products are facing great demand today. To meet this demand, increase quality and enlarge the range of applications, new second generation systems are emerging. The second generation will be characterized by digital speech transmission and enhanced capabilities of wireless

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

terminals to exchange signaling and control information with the remainder of the network. But at the same time, researchers are increasing their attention to third generation technologies to expand the limits further.

At present, there are about six different incompatible standards of cellular products used in different parts of the world. All of these products rely on analog frequency modulation for speech transmission and in-band signaling to move control information between terminals and the rest of the network during a call. Second generation cellular standard will conform with at least three different standards: one for Western Europe - Groupe Special Mobile (GSM), one for North America (IS-54), and another standard for Japan. While the European and Japanese standards will be applied to completely new cellular systems operating in dedicated frequency bands, the North American standard specified dual-mode operation. The standard IS-54 while accommodating the first generation standard, Advanced Mobile Phone System (AMPS), also adds a digital transmission capability to new subscriber equipments. In addition, in North America there are three more dual-mode enhancements in progress upon AMPS. Thus while Europe will migrate to a new digital system, North American cellular system will deploy various dual-mode technologies, all sharing a common analog standard, but with different digital advanced technologies. Cordless phones, are stand-alone consumer products are simple products where each comes with its own base station and needs compatibility only with its own base. But the second generation cordless phones will be designed as components of a very large network. Products that conform to a British standard, Cordless Telecommunications, second generation (CT2), or a Pan-European standard, Digital European Cordless Telecommunications (DECT), will have access to public base stations at telepoints and also to base stations installed as components of private branch exchanges in business environments. Security and privacy will be established through registration and handshaking procedures.

3.8.1 Cellular network architecture

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

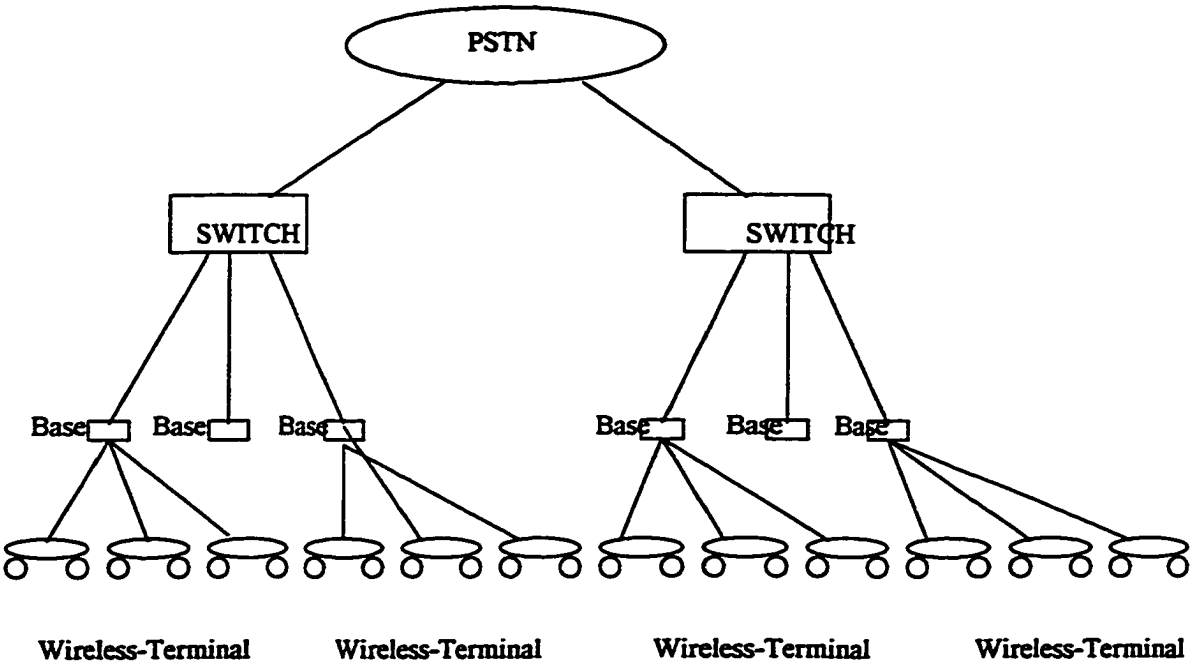


Figure 117:Cellular Network Architecture

Compared to current systems, the second generation cellular systems will have more open interfaces between network elements. But the basic network elements will remain the same. The elements required for providing wireless access to a fixed information network are: wireless-terminals, base stations and switches as shown in fig 117.

A base station exchanges radio signals with the wireless terminals and a switch connects the wireless network to a fixed network, the Public Switched Telecommunications Network (PSTN). Each base station serves dozens of wireless terminals simultaneously and a switch connects the user signals from upto 100 base stations to trunks of the public telephone network. In addition to making these connections, the cellular switch controls the assignment of radio channels to wireless terminals.

To serve the roamers (subscribers in locations remote from their home service areas), a number of proprietary communication links between cellular switches have been established. The trend in second generation cellular systems is to manage roaming and intersystem handoff with standardized signaling linking cellular switches and databases. The architecture of the Pan-European digital cellular system

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

called GSM is shown in fig 118 which in addition to the above three elements need three more elements: three databases. Home Location register, Visitor Location Register, Equipment Identity Register.

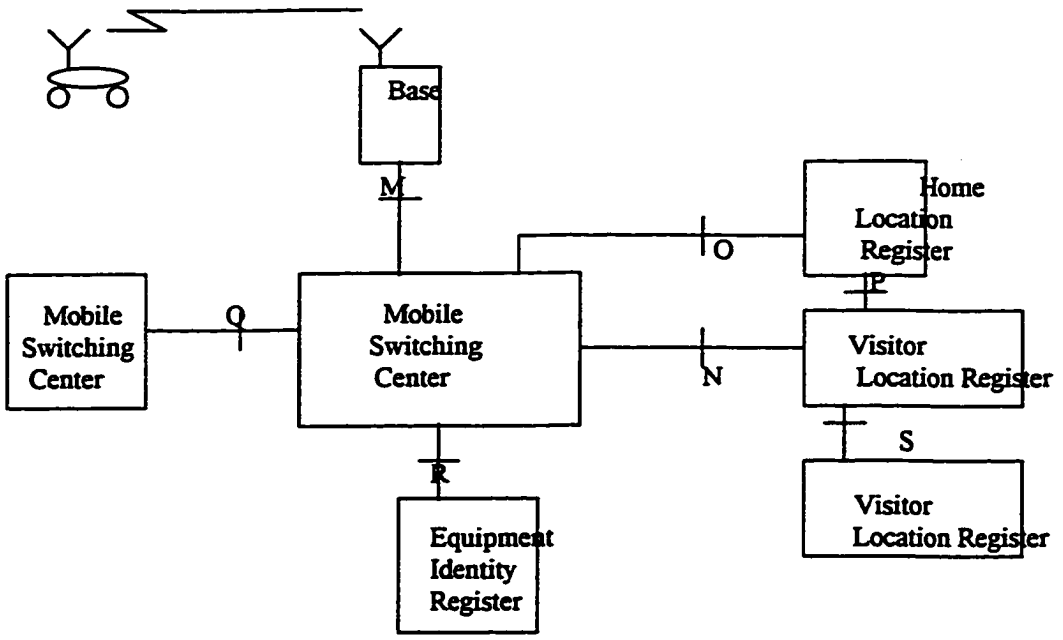


Figure 118:GSM Cellular network configuration

In the GSM networks all the interfaces among the elements are standard, but in the cellular system IS-54, the base station to switch interface is proprietary and work is in progress to define a standard interface between cellular switching offices to facilitate roaming and intersystem handoff. One important new thing about IS-54 is that the network control functions are much more distributed between the switches, base stations and mobile terminals. One important feature of both IS-54 and GSM digital cellular is mobile assisted handoff. In first generation cellular, the system controller in the mobile switching office assembles information from several base stations about the strength of the signal path from each active mobile terminal. On the basis of these measurements, the controller decides when the terminal should tune to a channel at a different base station. With the number of subscribers and the number of cells increasing rapidly, the volume of handoffs is approaching the processing capacity of the controllers. The second generation cellular telephones will measure the strength of signals received from neighboring base stations. These measurements will be reported to the serving base station, which will ascertain the need of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

a handoff and communicate to the system controller at the switching office the preferred new cell. Such an offloading approach will avoid overloading the controller and facilitate faster handoff.

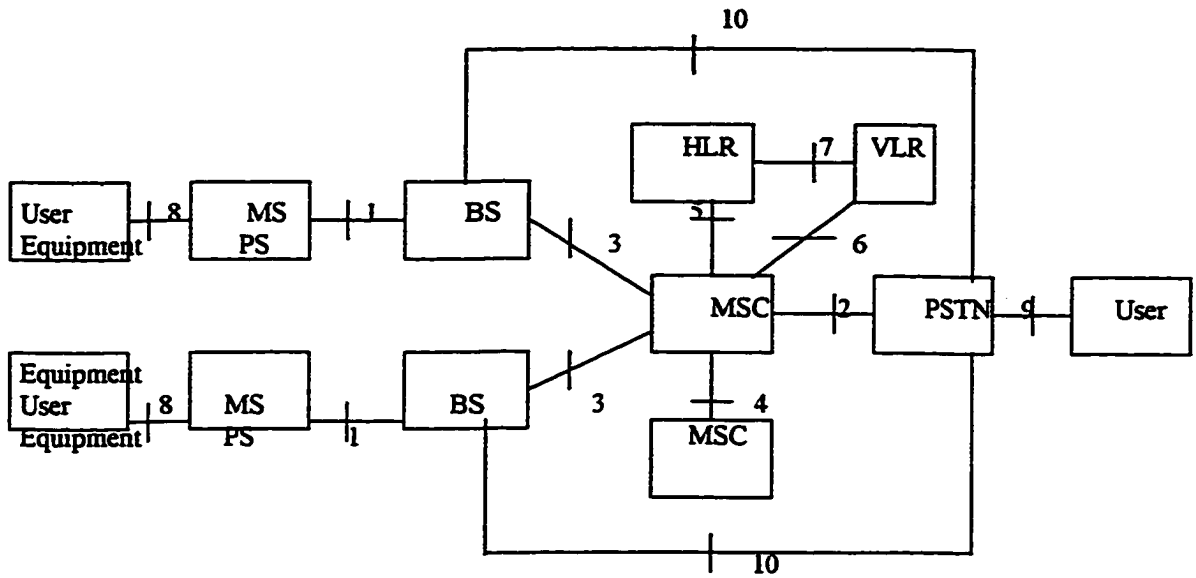
3.8.2 Cordless Network architecture

Viewed as a wireless information network, a first generation cordless phone is as simple as it can be. A single base station communicates with one and only one wireless terminal, and the switch is merely a make or break connection between the base station and the 2-wire subscriber line. Channel selection is solely the task of each cordless telephone. The multi-channel phones provides the option of switching from one channel to another one in case there is much cross-talk to the user. With the arrival of second-generation technology, cordless telephones will change from stand-alone item to elements of a geographically dispersed network. Network control will remain distributed, with wireless terminals automatically adjusting the channels. The owners will subscribe to the public telepoint services and use the cordless phones at thousands of locations with base stations connected to subscriber lines of the public telephone network. In addition, the same instruments will operate in business environments where cordless telephone technology is incorporated in wireless private branch exchanges. In large business premises, cordless telephones will have access to several base stations that provide intercell handoff as a user moves from one location to another one during a call.

3.8.3 Third generation Architecture

CCIR Task Group 8/1, in its work on "Future Land Public Mobile Telecommunications Systems", has developed the functional reference model of fig 119, which is a slight modification on GSM model of Fig 118.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



BS: Base Station
HLR: Home location Register
VLR: Visitor Location Register
MSC: Mobile switching Center

Figure 119:Future public land mobile telecommunications system model

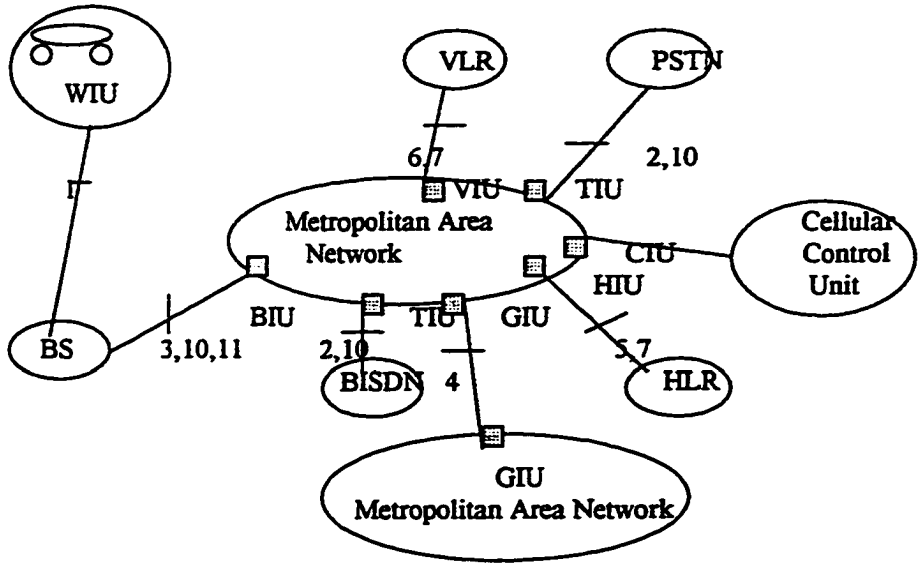
Here the wireless terminal is designated Mobile Station (MS) and portable Station (PS) and a standard interface (number 8) links it to subscriber equipment such as a telephone, a computer or a facsimile machine.

Anticipating large demand for access to future wireless networks, researchers have studied a cellular packet switch to implement the functions in fig 119. Packet communications in this application will serve to:

- Reduce substantially the burden on the central controller of radio resource management.
- Efficiently multiplex speech, network control information, and a wide variety of user data transmissions.
- Harmonize with the anticipated BISDN which will be based on packetized asynchronous transfer mode communications.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In the cellular packet switch in fig 120, all of the base stations in a service area are connected to a metropolitan area network which is also connected to trunks of the public switched telephone network and facilities of a BISDN.



BS: Base station
VLR: Visitor Location Register
HLR: Home Location Register

Figure 120:Cellular packet switch

Network interface units associated with each wireless terminal interface unit (WIU), central office trunk interface unit (TIU), and Base station Interface Unit (BIU) contain simple routing tables. Based on the information in these routing tables, which changes dynamically, the interface units direct each information packet to the correct destination even as users move frequently between base station service areas. All of this is possible because each packet contains, in addition to user information or network control information, address fields indicating the origin and destination of the packet. The routing is based on a combination of virtual circuit packet switching and datagram switching. The main call-processing function of the cellular controller is to assign a VCI to a wireless terminal and a PSTN or BISDN trunk at the beginning of each call. Other interfaces to the MAN include a Visitor location register Interface Unit (VIU), a Home location register Interface Unit (HIU), and a Gateway Interface Unit (GIU)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

for connection to another cellular packet switch. This implements all the CCIR interfaces as in fig 119 and a new interface 11 that links base stations directly. This interface is valuable in providing seamless, mobile-controlled handoff with no loss of information as terminals change cells.

3.8.4 Transmission Technologies

In all first generation cellular and cordless technology products, uses analog FM over radio channels within frequency bandwidth 12.5-30 kHz. Frequency division multiple access separates the signals of different terminals and simultaneous two-way communications are based on frequency division duplex. The second generation networks, using digital transmission will employ a variety of access technologies including narrow-band time division with eight channels per carrier (GSM), narrow-band time division with three channels per carrier (IS-54), frequency division (CT2) and time division with twelve channels per carrier (DECT). Continuous two-way communications take place in the two second generation cellular systems using frequency division duplex with 45 MHz spacing between the upstream and downstream carriers. Both cordless systems rely on time division duplex for two-way communications.

3.8.4.1 GSM: Pan-European Digital Cellular

All four systems use burst-mode mobile-to-base transmission based on slot and frame structures with varying degrees of complexity. The timing patterns of GSM are by far the most intricate, with time intervals defined as short as 920 ns (one quarter a bit) and as long as 3h 28 min 53.76 s (encryption hyper-frame). The GSM multiple access technique is a combination of frequency division with carriers spaced at 200 kHz intervals and time division with eight logical channels per carrier. During a call, each terminal has access to a two-way digital traffic channel and a separate two-way control channel.

Figs 121 and 122 show how these channels are organized. Fig 121 shows a GSM multiframe of 120 ms duration divided into 26 frames, each containing eight time-slots. In the multi-frame, 24 frames are for user information carried in logical traffic channels. The other two frames carry system control information in associated control channels. Each call in progress is assigned one of these logical control

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

channels which provide GSM with an out-of-band signaling capacity which does not exist in first generation cellular systems. Fig 122 shows the composition of a GSM time slot. Two bursts of 58 bits account for most of the transmission time in a slot. Fifty seven data bits carry user information and the other bit is a flag used to distinguish speech from other transmissions. There is an equalizer training sequence consisting of 26 b in the middle of a time-slot, and the slot begins and ends with three "tail-bits", all logical zeroes.

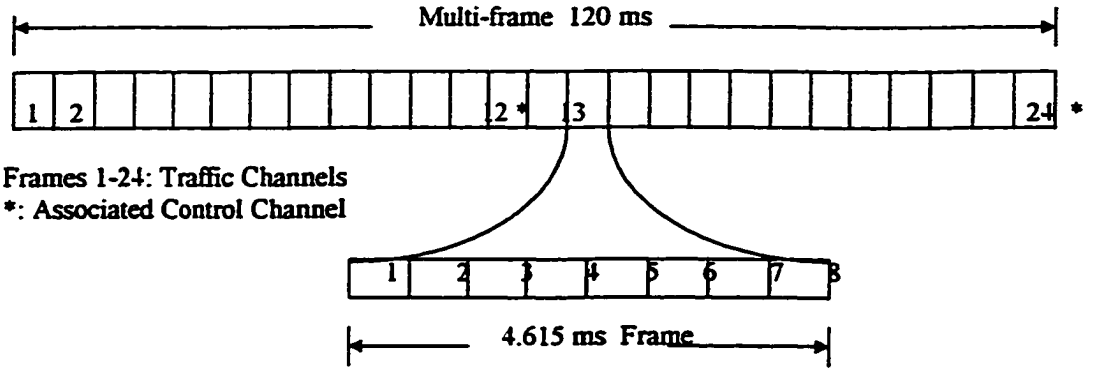
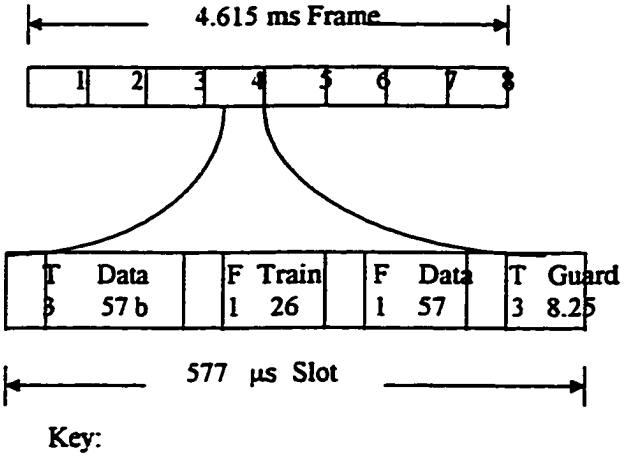


Figure 121:GSM multi-frame and frame structure

The guard time of 8.25 b intervals prevents overlap at a base station of signal bursts arriving from different terminals. With 156.25 b/577 μs, the transmission rate is 270.833 kb/s.



Key:

T: Tail bits
F: Flag
Train: Equalizer Training Sequence

Figure 122:GSM time slot structure

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

At this rate, an adaptive equalizer is essential in the multi-path transmission environment of a cellular system. The bit interval is 3.69 μs and GSM mandates that receivers operate accurately when multiple signal paths have delay differences upto 16 μs, which is more than four bit intervals.

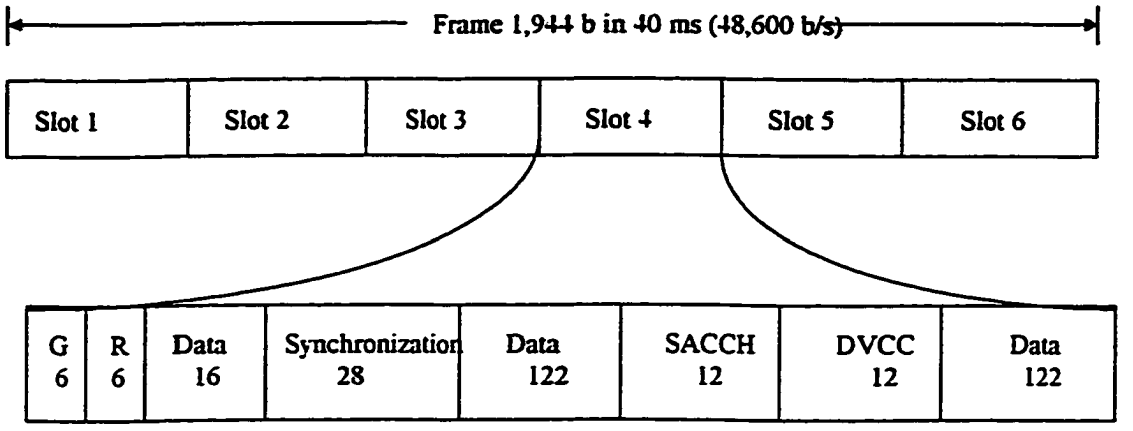
The GSM modulation is Gaussian minimum shift keying, in which the modulator bandpass filter has a 3 db cutoff frequency of 81.25 kHz (0.3 times the bit rate). The modulation efficiency of 271 kb/s signals operating with a 200 kHz carrier spacing is 1.35 b/s/Hz. The GSM speech coder is referred to as linear predictive coding with regular pulse excitation. The source rate is 13 kb/s and the transmission rate, including error detecting and correcting codes, is 22.8 kb/s.

In early GSM implementations, each full rate traffic channel will occupy all 24 frames in the multi-frame of fig 121. The associated control channel will be carried in one other frame. Later implementations will add spectrum efficiency by reducing the allocation to 12 frames per half rate channel per multi-frame. The speech transmission rate will be 11.4 kb/s and a carrier will have a capacity of 16 user channels. Then both associated control channel frames in fig 121 will be occupied.

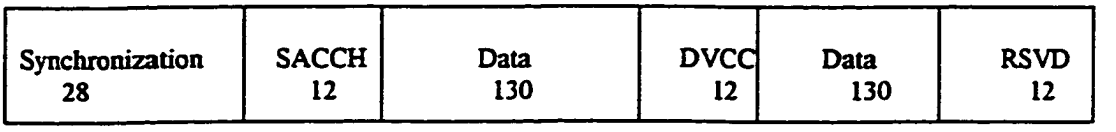
IS-54 North American Digital Cellular

The carrier spacing of IS-54 is 30 kHz, as in the first generation AMPS system. Operating companies will selectively convert analog channels to digital operation in order to relieve traffic congestion at cellular base stations. Each digital channel operates at 48.6 kb/s carrying three user signals. The IS-54 timing structure, shown in fig 123 is simpler than that of GSM. The frame duration is 40 ms divided into six 6.67 ms time slots. One time slot carries 324 b, including 260 b of user information (data in fig 123) and 12 b of system control information (Slow Associated Control Channel - SACCH). The remaining 52 b carry a time synchronization signal (28 b), a digital verification color code (12 b), and in the mobile-to-base direction a 6 b guard time interval, when no energy is transmitted, followed by a 6 b ramp interval to allow the transmitter to reach a full output power level.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Mobile to Base



Base to Mobile

G: Guard Time
R: Ramp Time
DVCC: Digital Verification Color Code
RSVD: Reserved for future use

Figure 123:IS-54 slots and frames

Table 3-5: PRMA variables

Channel Rate	720 kb/s
Source Rate	32 kb/s
Frame Duration	16 ms
Overhead	64 b/packet
Maximum Delay	32 ms

Like GSM, IS-54 defines full rate and half rate logical traffic channels. Initial IS-54 implementations will include only full rate channels, which occupy two slots per 40 ms frame. With 260 b of user information

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

per slot, full rate channels operate at 13 kb/s. Eventually, IS-54 will introduce 6.5 kb/s half rate channels that occupy one slot per frame.

The 28 b synchronization field contains a known bit pattern that allows the receiver to establish bit synchronism and to train an adaptive equalizer. The system specifies six different synchronization patterns, one for each slot in the 40 ms frame. This allows the receiver to lock onto its assigned time slots. The digital verification color code plays the role of the supervisory audio tone in the analog AMPS system. There are 256 8b color codes, protected by a (12;8;3) Hamming code. Each base station is assigned one of these codes and the verification procedure prevents a receiver from locking onto an interfering signal from a distant cell.

In a departure from the constant envelope modulations of first generation wireless access systems and other second generation systems, IS-54 adopts a linear modulation technique, Differential Quaternary Phase Shift Keying (DQPSK). The principal advantage of this modulation method is bandwidth efficiency. The transmission rate is 48.6 kb/s with a channel spacing of 30 kHz, which comes to 1.62 b/s/Hz, a 20% improvement over GSM. The main penalty of linear modulation is power efficiency, which affects the weight of hand held portable terminals and time between battery chargings. The specific method of modulation is $\pi/4$ shifted DQPSK with root cosine rolloff filtering at the transmitter and receiver. The rolloff factor is 0.35, which places a zero at 16.4 kHz in the baseband spectrum of the transmitted signal. The IS-54 speech coder is a type of codebook excited linear predictive coding referred to as Vector Sum Excited Linear Prediction (VSELP). The source rate is 7.95 kb/s and the transmission rate is 13 kb/s.

CT2

Of the four major second generation wireless information networks, CT2 is the only one that uses frequency division multiple access. Carriers spaced at 100 kHz each convey one conversation, using time division duplex for two-way transmission. The bit-rate of CT2 is 72 kb/s and its timing structure, shown in fig 124, is the simplest system. A frame of 2 ms duration contains two time slots, one for portable to fixed

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

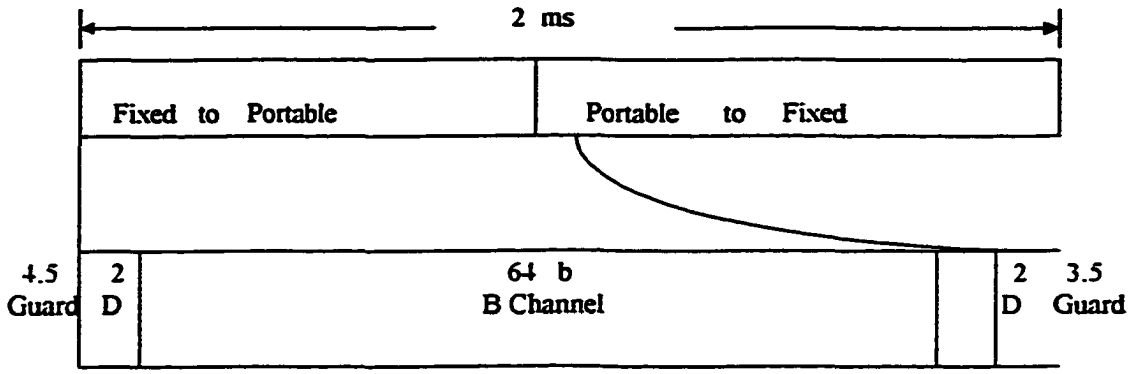


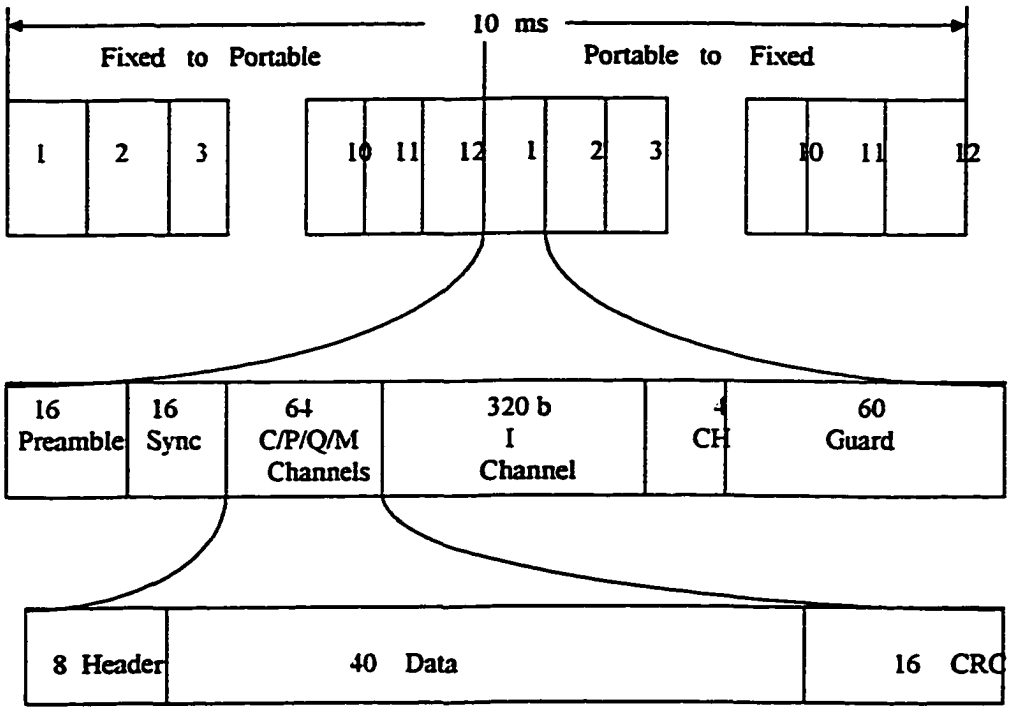
Figure 124:CT2 slot and frame

station (mobile-to-base) transmission and the other for fixed-to-portable. Within each slot, there are 64 b for user information and 4 b for system control information. (The specification also admits an alternate multiplex arrangement with only 2 b of control information per slot and correspondingly longer guard times between slots.) This comes to 136 b/frame, to which CT2 adds two guard times of 8 b total duration for an aggregate of 144 b/2ms frame. The CT2 modulation technique is binary frequency shift keying. With a channel spacing of 100kHz, the bandwidth efficiency of CT2 is 0.72 b/s/Hz, approximately half of that of GSM. The speech coder is a standard adaptive differential pulse code modulator operating at 32 kb/s.

DECT

Ten carriers, each occupying a bandwidth of 1.728 MHZ, will each carry 12 channels in a Time Division Multiple Access (TDMA) format. DECT, in common with CT2, employs time division duplex so that information moves in both directions over the same carrier. As shown in fig 125, the frame duration is 10 ms, with 5 ms for portable-to-fixed station transmission and 5 ms for fixed-to-portable. The transmitter

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



CRC: Cyclic Redundancy Check
Ch: Parity Check bit
Sync: Synchronization

Figure 125:DECT slot and frame

assembles information in signal bursts which it transmits in slots of duration 10/24 = 0.417 ms. With 480 b/slot (including a 64 b guard time), the total bit rate is 1.152 Mb/s, the highest by far of any of the second generation systems. Each slot contains 64 b for system control (C, P, Q and M channels) and 320 b for user information (I channel). In common with GSM, the modulation method of DECT is Gaussian minimum shift keying. However, the relative bandwidth (BT) of the Gaussian filter is wider (0.5 times the bit rate) than in GSM (BT=0.3). The bandwidth efficiency, 1.152 Mb/s in 1.728 MHZ or 0.67 b/s/Hz, is comparable to that of CT2. In common with CT2, the speech coder of DECT is adaptive differential pulse code modulation with a bit rate of 32 kb/s. The transmission methods of the two cordless systems reflect the emphasis on simple terminals and signal quality comparable to conventional telephones. The relative complexity of the cellular systems derives from the importance of bandwidth efficiency and from the difficulty of achieving reliable information transfer in a hostile environment.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Control Channels

For call setup, cellular systems rely on broadcast control channels (base-to-mobile) and shared memory access control channels (mobile-to-base). While a call is in progress, the system interrupts the user voice signal (typically for 100 ms) when it is necessary to send a control message to or from the cellular phone. This blank-and-burst (in band) mode of transmission is used primarily for power control and handoff. The interruptions of the voice channel produce audible clicks, which severely impair communication quality if they occur too frequently. As a consequence, first generation cellular networks have a limited capacity to move network control information between terminals and base stations during a call.

First generation cordless phones use product-specific techniques for transmitting control information. Except for in Japan, all cordless telephone control information moves through the voice channel. Cordless phones with a capability of transmitting or receiving control information during a call use the blank-and-burst technique. By contrast, all second generation cordless and cellular systems will incorporate dedicated channels for moving control messages to and from wireless terminals without interrupting user information. These channels will enhance the wireless networks themselves and greatly increase their capability to deliver to wireless terminals new features of the ISDN public network.

Cellular

In both of the second generation cellular systems, the dedicated control channels are referred to as Associated Control Channel (ACCH). In each system there is a SACCH that operates throughout a call and a fast ACCH (FACCH) used primarily for handoff. Figures 121 and 123 show how the SACCH is multiplexed with user information in GSM and IS-54 respectively. In GSM, two frames of every 120 ms multi-frame are reserved for SACCH information. In early implementations, with eight full rate channels per carrier, only one of the two SACCH frames will be occupied. Eventually, GSM will operate with 16 half rate channels per carrier, and they will use all 16 slots in the two SACCH frame. A SACCH message contains 184 information bits. Error control coders process this information to produce 456 channel bits which are interleaved and distributed over four time slots. With one time slot per multi-frame and a multi-

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

frame duration of 120 ms, the total transmission time of a SACCH message is 480 ms. Thus the transmission bit rate is 456 b in 480 ms, or 950 b/s. The information rate is 184 b in 480 ms, or 383 b/s.

In IS-54, a control channel message has a length of 66 b, which in the SACCH are protected by a rate ½ forward error correcting coder that generates 132 b per message. This information is interleaved over 12 time slots and transmitted I the SACCH fields as shown in fig 123. In a system operating with full rate digital traffic channels, the total delay (12 slots) of the SACCH is six IS-54 frames, or 240 ms, and the information bit rate is 300 b/s. With half rate channels, the SACCH rate information rate will be 150 b/s and the message delay will be 480 ms. In both GSM and IS-54, the FACCH preempts user information frames to achieve rapid handoff. While this mode of operation is a return to the blank-and-burst technique of the first generation, both systems use signal processing to estimate the speech that was deleted to make room for a FACCH message.

Cordless

Figures 124 and 125 show that as in IS-54, each burst of user information in CT2 and DECT is accompanied by system control information. The CT2 control channels are referred to as D channels. As shown in fig 124, each time slot contains four D-channel bits for a total bit rate of 2 kb/s in each direction (or 1 kb/s with terminals that operate with two, rather than four, D-channel bits per time slot). Control messages consist of one to six 64 b code words. A code word is distributed over 16 frames and, therefore has a delivery time of 32 ms. Each code word is protected by 16 error-detecting parity bits. The receiver acknowledges code words that satisfy all parity checks. Upon detecting transmission errors, the receiver requests retransmission of the affected code word. With 48 information bits per code word, the maximum information throughput (in the absence of retransmission) of the CT2 control channel is 3 kb/s, far in excess of the signaling capability of the second generation cellular systems.

DECT control information is carried by 64 b in every time slot of an established communication. These bits are assigned to one of four logical channels (either C, P, Q or M channel) depending on the nature of the control information. Thus the raw data rate is 6.4 kb/s. As in CT2, DECT relies on error detection and retransmission for accurate delivery of control information, rather than the forward error

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

correction of the cellular systems. Thus, every 64 b control, word contains 16 Cyclic Redundancy Check (CRC) bits in addition to the 48 b payload. The maximum information throughput of the DECT control channel is 4.8 kb/s. With each control word contained in a single frame the average delay is 5 ms. Clearly among the second generation systems surveyed so far, the DECT control channel has the best throughput and delay performance.

PCS network Architecture and call model

A personal communication services network is a wireless network where the service area of the network is populated with base stations. The radio coverage of a base station is a cell. The base station locates the subscriber and delivers calls to or from the portable by means of paging within the cell it serves. A registration area (RA) consists of an aggregation of cells, forming a contiguous geographical region. To connect a phone call to a roaming portable, it is necessary to identify the portable's RA. The strategies proposed are two-level hierarchies that maintain a system of home and visited databases (home location registers, or HLRs, and visitor location registers, or VLRs). To order PCS services, a PCS subscriber must "enroll" with a PCS provider and provide necessary informations as credit, service type and current location to set up the PCS account. These informations are stored in the HLR of the PCS provider. When the PCS subscriber roams to another RA, which is likely to be owned by another PCS provider, the PCS subscriber becomes a visitor of that RA. The VLR of the RA is used to store the visiting PCS subscriber's information. Upon registering with the VLR, the VLR notifies the HLR of the visiting PCS subscriber that "your subscriber is at my place". There are general modeling framework for the data management scheme. The framework includes the modeling of call and movement traffic, data management activities and the signaling network. The framework can be divided into the following components: Topology model, Call model and Movement model. The topology model describes the geographical topology or the connectivities between sites. The call model and movement model describe the patterns how users make calls to each other and how they move through the geography defined by the topology model.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The call model can be divided into two parts: the call model that describes how often individual users place calls to other users and characterizes the duration of each call, the callee distribution model which characterizes the callee. Normally a call traffic model generates call arrivals for different classes of traffic and models time-varying user behaviors. Each call traffic class is characterized by its probability of occurrence, call arrival rate, mean call duration and distribution. Also the call model describes time-of-day call traffic volume patterns for weekdays and weekends. The callee distribution model characterizes how the callee is generated for each call.

The mobility models conceptualize the traffic flow from one cell to another. There are many models available. Fluid model perceives traffic movement as a macroscopic behavior considering it analogous to flow of fluid. In its simplest form, the traffic flowing out of a region is proportional to the population density, the average velocity and the regional diameter. Thus the number of crossings per unit time, N is given as

$N=\pi \rho L v$ where v is the average velocity, ρ is the population density and L is the region diameter. A more sophisticated model can be built using the diffusion process.

Markovian model is a random-walk model describing individual movement behavior. In this model, a subscriber will either remain within a region or move to an adjacent region according to a transition probability distribution.

Gravity model has been extensively used in transportation research which likens the movement with gravitational pull between two regions of populations and it is proportional to the population sizes.

In addition, there are other models as metropolitan mobility model describing subscriber movements within a metropolitan area.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.9 Mobile IP

As number of hosts connected to the internet has been increasing, further growth is likely to be fueled by growth in notebook computers along with the growth in wireless communication devices. Such devices can only add more options for connecting to internet. There are many radio attachments and infrared devices: of course communications by way of cellular phone is also an option though costly. Mobile IP (an enhancement on IP) is the only current means for offering seamless roaming to mobile computers in the internet and its specification from IETF is RFC 2002.

Mobile IP can be thought of as the cooperation of three major subsystems. First, there is a discovery phase where the mobile computer can determine their new attachment points (new IP addresses) as they roam from place to place within the internet. Second, once the mobile computer knows the IP address at its new attachment point, it registers this care-of address with an agent at its home network. Lastly, mobile IP defines mechanisms to deliver datagrams to the mobile node when it is away from the home network. These three steps of mobile IP protocol are called : 1) Agent Discovery, 2) Registration and 3) Tunneling. Mobile IP extends IP by allowing the mobile computer to effectively utilize two IP addresses, one for identification and one for routing. Before getting into the details a few functional entities will need to be defined.

Mobile node

A host or router that changes its point of attachment from one network or subnetwork to another one, without changing its IP address. A mobile node can continue to communicate with other Internet nodes at any location using its constant IP address.

Foreign agent

A router on a mobile node's visited network that incorporates with the home agent to complete the delivery of datagrams with the home agent to complete the delivery of datagrams to the mobile node while it is away from home.

Home agent

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A router on a mobile node's home network that delivers datagrams to the off-site mobile nodes, and maintains current location information for each.

A mobile node has a home address, which is a long-term IP address on its home network. When away from its home network, a *care-of address* is associated with the mobile node which reflects the mobile node's current point of attachment. The mobile node uses its home address as the source address of all IP datagrams it sends, except where otherwise required for certain registration request datagrams.

Agent advertisement

Foreign agents advertise their presence by using a special message, which is constructed by attaching a special extension to a router advertisement.

Care-of address

The termination point of a tunnel toward a mobile node, for datagrams forwarded to the mobile node while it is away from home. There are two different types of care-of address: a foreign agent care-of address is an address of a foreign agent with which the mobile node is registered; a collocated care-of address is an externally obtained local address which the mobile node has associated with one of its own network interfaces.

Correspondent node

A peer with which a mobile node is communicating. A correspondent node may be either mobile or stationary.

Foreign network

Any network other than the home network which the mobile node is connected to.

Home address

An IP address that is assigned for an extended period of time to a mobile node. It remains unchanged regardless of where the node is attached to the internet.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Home network

A network, possibly virtual, having a network prefix matching that of a mobile node's home address. Note that standard IP routing mechanisms will deliver datagrams destined to a mobile node's home address to the mobile node's home network.

Link-Layer Address

The address used to identify an end-point of some communication over a physical link. Typically, the link-layer address is an interface's media access control (MAC) address.

Mobility agent

Either a home or a foreign agent.

Mobility binding

The association of a home address with a care-of address, along with the remaining lifetime of that association.

Mobility security association

A collection of security contexts between a pair of nodes which may be applied to mobile IP protocol messages exchanged between them. Each context indicates an authentication algorithm and mode, a secret (a shared key, or appropriate public/private key pair), and a style of replay protection in use.

Nonce

A randomly chosen value, different from previous choices, inserted in a message to protect against replays.

Security parameters index (SPI)

An index identifying a security context between a pair of nodes among the contexts available in the mobility security association.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Tunnel

The path followed by a datagram while it is encapsulated. The model is that, while encapsulated, a datagram is routed to a foreign agent, which decapsulates the datagram and then forwards it along its ultimate destination.

Virtual network

A network with no physical instantiation beyond its router (with a physical network interface on another network). The router (e.g., a home agent) generally advertises reachability to the virtual network using conventional routing protocols.

Visited network

A network other than the mobile host's home network to which the mobile node is currently connected.

Visitor list

The list of mobile nodes visiting a foreign agent.

3.9.1 Protocol Overview

The three steps in mobile IP as discussed before are the following:

1) **Agent Discovery** - The mobility agent advertise for their availability on each link for which they offer service.
2) **Registration** - When the mobile node is away from home, it *registers* its care-of address with its home agent.
3) **Tunneling** - In order for datagrams to be delivered to the mobile node when it is away from the home network, the home agent has to tunnel the datagrams to the care-of address.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A rough outline for mobile IP protocol has been presented making use of the three steps above. Fig 126 also shows the network entities involved in the two-way communication between an IP host and a mobile node over IP.

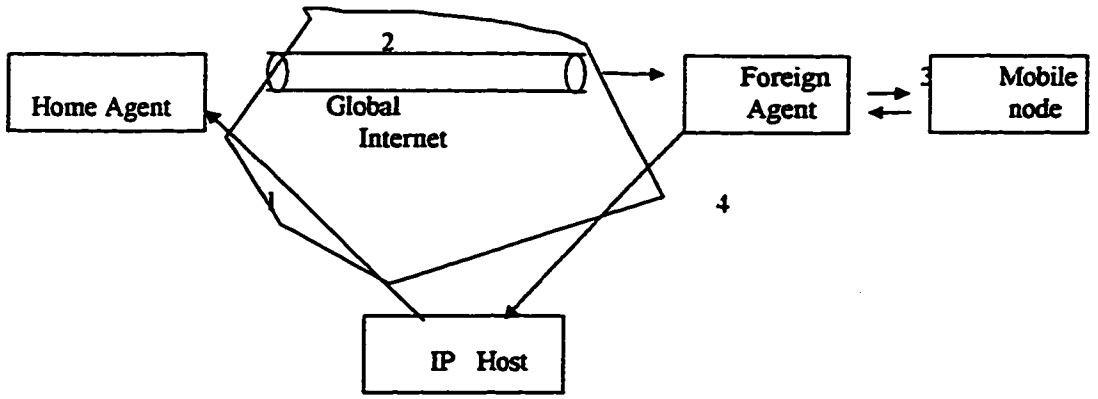


Figure 126:Mobile IP datagram flow

- Mobile agents make themselves known by sending agent advertisement messages. An impatient mobile node may optionally solicit an agent advertisement message.
- After receiving an agent advertisement, a mobile node determines whether it is on the home network or a foreign network. A mobile node basically works like any other node on its home network when it is at home.
- When a mobile node moves away from its home network, it obtains a care-of address on the foreign network, for instance, by soliciting or listening for agent advertisements, or contacting Dynamic Host Configuration Protocol (DHCP) or Point-to-Point Protocol.
- While away from home, the mobile node registers each new care-of address with its home agent, possibly by way of a foreign agent.
- Datagrams sent to the mobile node's home address are intercepted by its home agent, tunneled by its home address to the care-of address, received at the tunnel endpoint (at either a foreign agent or the mobile node itself), and finally delivered to the mobile node.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- In the reverse direction, datagrams sent by the mobile node are generally delivered to their destination using standard IP routing mechanisms, not necessarily passing through the home agent.

When the home agent tunnels a datagram to the care-of address, the inner IP header destination (i.e. the mobile node's home address) is effectively shielded from intervening routers between its home network and its current location. At the care-of address, the original datagram exits from the tunnel and is delivered to the mobile node.

It is the job of every home agent to attract and intercept datagrams that are destined to the home address of any its registered mobile nodes. The home agent basically does this by using a minor variation on proxy Address Resolution Protocol (ARP), and to do so in the natural model it has to have a network interface on the link indicated by the mobile node's home address. However, the latter requirement is not part of the mobile IP specification. When foreign agents are in use, similarly, the natural model of operation suggests that the mobile node be able to establish a link to its foreign agent. Other configurations are possible, however, using protocol operations, not defined by mobile IP. Notice that, if the home agent is the only router advertising reachability to the home network, but there is no physical link instantiating the home network, then all datagrams transmitted to mobile nodes addressed on that home network will naturally reach the home agent without any special link operations.

Figure 126 illustrates the routing of datagrams to and from a mobile node away from home, once the mobile node has registered with its home agent. The mobile node is presumed to be using a care-of address provided by the foreign agent.

- A datagram to the mobile node arrives on the home network via standard IP routing.
- The datagram is intercepted by the home agent and is tunneled to the care-of address, as depicted by the arrow going through the tube.
- The datagram is detunneled and delivered to the mobile node.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- For datagrams sent by mobile node, standard IP routing delivers each to the destination. In the figure the foreign agent is the mobile node's default router.

3.9.1.1 Mobile agent Discovery

The process of detecting a mobile agent is quite similar to that used by Internet nodes to detect routers running Internet Control Message Protocol (ICMP) Router Discovery (RFC 1256). The basic operation involves periodic broadcasts of advertisements by the routers onto their directly attached subnetworks. Noticing the similarity, the Mobile IP working group decided to use RFC 1256 directly and support the special additional needs of mobility agents by attaching special extensions to the standard ICMP messages.

Agent Advertisement

By far the most important extension is the mobility agent extension, which is applied to ICMP Router Advertisement and shown in fig 127.

The flags R, B, H, F, M, G, and V inform mobile nodes regarding special features of the advertisement, and are described here. The type field allows mobile nodes to distinguish between the various kinds of extensions that may be applied by mobility agents to the ICMP Router Advertisements; the type for the mobility agent advertisement extension is 3. Other extensions may, of course, precede or succeed this extension; almost no other extensions are defined as known of yet. The length field is the length of this single extension, which really only depends on how many care-of addresses are being advertised. Furthermore, currently at most one care-of address will typically be advertised. Home agents do not have to advertise care-of addresses, but they still need to broadcast mobility agent advertisements so that mobile nodes will know when they have returned to their own home network. Indeed, mobility agents can advertise care-of addresses even when they do not offer any default router addresses, as would be found in other ICMP Router Advertisements. No *preferences* apply to advertised care-of addresses. The flags are defined as follows:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

R Registration required. Registration with this foreign agent (or another foreign agent on this link) is required, even if using a collocated care-of address.

B The foreign agent is busy.

H The agent is a home agent.

F The agent is a foreign agent

M Minimal encapsulation (RFC 2004)

G GRE encapsulation (RFC 1701)

V Van Jacobson header compression (RFC 1144)

Note that bits F and H are not mutually exclusive, and that B can not be set until F is also set. Note also that a foreign agent typically needs to continue sending advertisements out (with the B bit set), even though it is too busy to offer service to new mobile nodes. Otherwise, the foreign agent's current customers might think that the foreign agent has crashed, and move away unnecessarily.

The mobility agent generally increments the sequence number by one for each successive advertisement. Special rules enable a mobile node to distinguish between foreign agent crashes, and wraparound of the sequence number field.

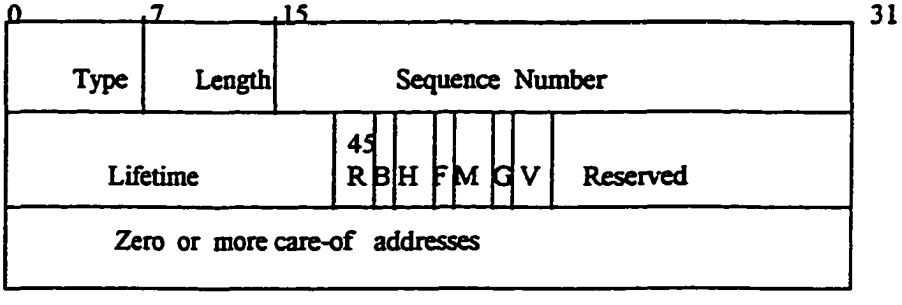


Figure 127:Mobility Agent extension format

Agent solicitation

A mobile node is allowed to send ICMP Router Solicitation messages in order to elicit a mobility agent advertisement.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.9.1.2 Registration

There are two kinds of registration messages, the registration request and registration reply, both sent to the User Datagram Protocol (UDP) port 434. The overall data structure of the registration messages is shown in fig 128. The request message allows the mobile node to inform its home agent of its current care-of address, tells the home agent how long the mobile node wants to use the care-of address, and indicates special features that may be available from the foreign agent.

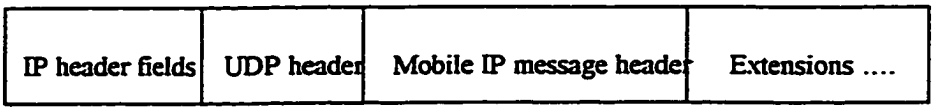


Figure 128:Data structure of a registration request

The foreign agent is considered a passive agent in the registration procedure, and agrees to pass the request to the home agent, and subsequently to pass the reply from the home agent back to the mobile node.

Registration Request

The registration process is the same irrespective of whether the mobile node had acquired its care-of address from a foreign agent or a DHCP server. In the former case, the mobile node basically sends the request to the foreign agent, which then relays the request to the home agent. In the latter case, the mobile node sends the request directly to the home agent, using the *collocated* care-of address as the source IP address of the request. After the IP and UDP headers, the registration request has the structure as in fig 129.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

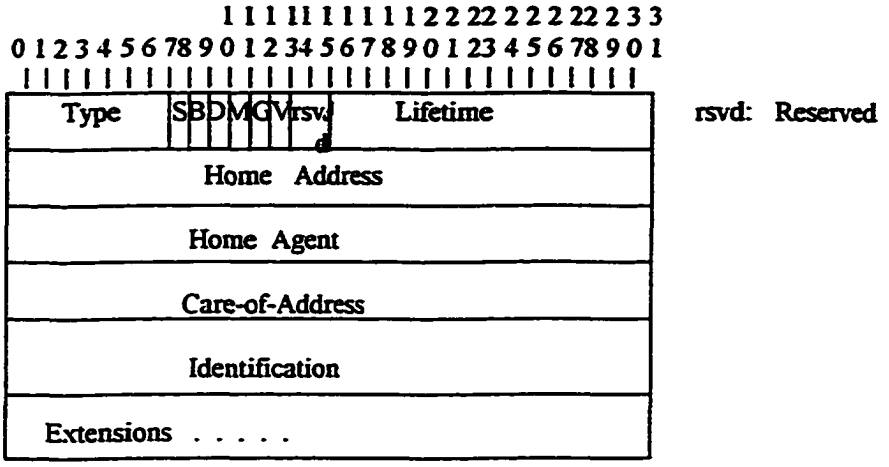


Figure 129:Registration Request Format

Given the discussion about the bit fields in the agent advertisement extension, the need for most of the fields is clear. The V bit in the request serves to inform the foreign agent whether Van Jacobson compression is required or not. The M and G bits tell the home agent which additional encapsulation methods can be used. The B bit is used to tell the home agent to encapsulate broadcast datagrams from the home network for delivery to the care-of address (and from there to the mobile node). The D bit describes whether or not the mobile node is collocated with its care-of address, and is mainly useful for determining how to deliver broadcast and multicast datagrams to the mobile node.

Also included are the home address and the proposed care-of address. The identification field, a 64-bit field is used for replay protection, as described below when security is discussed. The most important extension is the mobile-home authentication extension, described before, which is required in every registration in order to allow the home agent to prevent fraudulent remote redirects.

Registration Reply

The registration reply structure is in fig 130.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

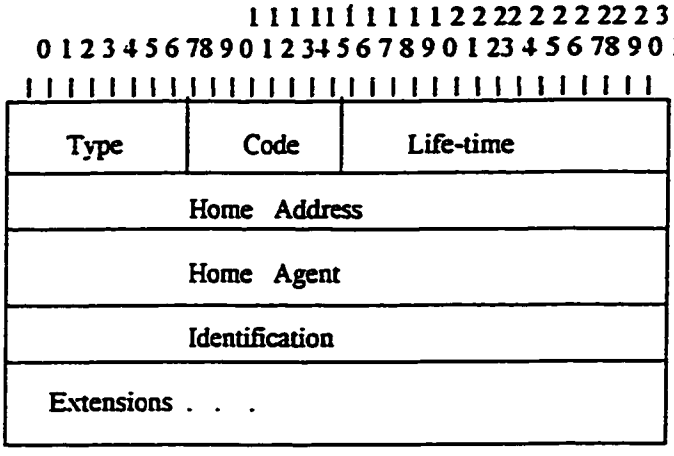


Figure 130:Registration reply format

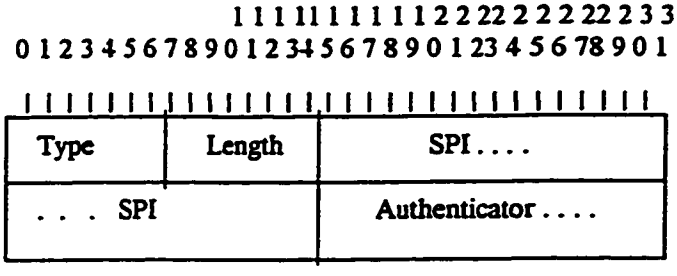


Figure 131:Mobile IP authentication extensions

The lifetime field tells the mobile node how long the registration will be honored by the home agent. It can be shorter than requested, but never longer. The code field describes the status of the registration. If the registration succeeds, it is well and good. If it fails, the code field offers details about what went wrong.

Typical values include:

0 Registration accepted

Registration denied by the foreign agent:

66 Insufficient resources

69 lifetime request > advertised limit

70 poorly formed request

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

71 poorly formed reply

88 home agent unreachable

Registration denied by the home agent:

130 Insufficient resources

131 Mobile node failed authentication

133 Registration identification mismatch

134 Poorly formed request

136 Unknown home agent address

Receiving code 133 usually indicates the need for resynchronization between the home agent and the mobile node. This synchronization can be either time-based or based on the exchange of randomly generated nonce values. Note that error code 130 should effectively be impossible. The home agent should not be configured to accept the mobile node if it does not have the required resources. Upto-date values of the code field can be found in any latest literature.

Dynamic Home Agent Discovery

Rejection code 136 forms the basis for allowing the mobile node to find the address of a home agent when needed. If the registration reply is addressed to the *directed broadcast* address, each home agent on the home network should receive and reject it. However, the registration reply containing the rejection also contains the home agent's address, so the mobile node can try again and succeed.

Securing the Registration Procedure

Registration in mobile IP must be made secure so that fraudulent registrations can be detected and rejected. Otherwise, any malicious user in the internet could disrupt communications between the home agent and the mobile node by the simple expedient of supplying a registration request containing a bogus care-of address. This would disrupt all traffic destined for the mobile node.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The method specified to protect against such malicious users involves the inclusion of an unforgeable value along with the registration that changes for each new registration. In order to make each registration number different, a random number or timestamp (a nonce) is inserted into the identification field. The home agent and mobile node have to agree on reasonable values for the timestamp or nonce, and the protocol allows for resynchronization, as described earlier by use of reply code 133.

There are three authentication extensions defined for use with mobile IP, as follows:

- the mobile-home authentication extension
- the mobile-foreign authentication extension
- the foreign-home authentication extension

As illustrated in fig 131, they all have similar formats, distinguishable only by different type numbers. The mobile-home authentication extension is required in all registration requests and replies. The SPI within any of the authentication extensions defines the security context used to compute (and check) the authenticator. In particular, the SPI selects the authentication algorithm and mode, and secret (a shared key or public/private key pair) used to compute the authenticator. A mobile node has to be able to associate arbitrary SPI values with any authentication algorithm and mode it implements. SPI values 0 through 255 are reserved and not allowed to be used in any mobility security association.

The default authentication algorithm uses keyed-MD5 in prefix+suffix mode to compute a 128 bit message digest of the registration message. The default authenticator is a 128-bit message digest computed by the default algorithm over the following stream bytes:

- The shared secret defined by the mobility security association between the node and by SPI value specified in the authentication extension, followed by
- The protected fields from the registration message, in the order specified above, followed by
- The shared secret again

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The authenticator itself and the UDP header are not included I the computation of the default authenticator value. All implementations of mobile IP are required to implement to default authentication algorithm just described.

3.9.1.3 Routing and Tunneling

The home agent, after a successful registration, will begin to attract datagrams bound for the mobile node and then *tunnel* each datagram to the mobile node at its care-of address. The tunneling can be done by one of several encapsulation algorithms, but the default algorithm that must be supported always is simple IP-within-IP encapsulation, as described on RFC 2003. Encapsulation is a very general technique used for various reasons, including multicast, multiprotocol operations, authentication, privacy, defeating traffic analysis and general policy routing.

Fig 132 shows how an IP datagram is encapsulated by adding a new IP header called the tunnel header.

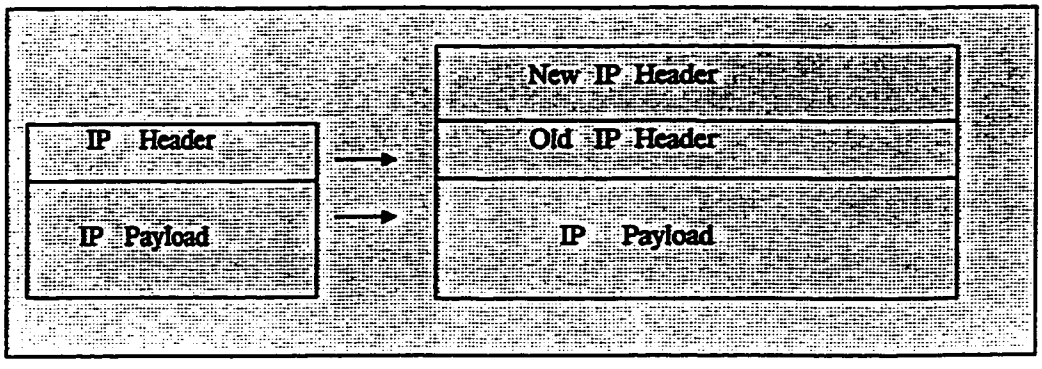


Figure 132:IP-within-IP encapsulation

In the case of mobile IP, the values of the fields in the new header are selected naturally, with the care-of address used as the destination IP address in the tunnel header. The encapsulating IP header indicates the presence of the encapsulated IP datagram by using the value 4 in the outer protocol field. The inner header is not modified except to decrement the TTL by 1.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

As an alternate, *minimal encapsulation* can be used as long as the mobile node, home agent and foreign agent all agree to do so. IP-within-IP uses a few extra bytes per datagram than minimal encapsulation method. but it allows fragmentation at the home agent when needed to pass through tunnels with smaller path MTUs. The minimal encapsulation header fits in the same relative location within the encapsulated payload, as indicated by the old IP header in fig 132. The presence of the minimal encapsulation header is identified by protocol number 55 in the new IP header protocol field. Fig 133 shows the fields of the minimal encapsulation header as described below. The length of the minimal header is either 12 or 8 depending on whether the original source IP address is present or not.

Minimal encapsulation format

Protocol: Copied from the protocol field in the original IP header.

Original Source Address Present : If 1, the original source address is present; otherwise not.

Reserved: Sent as zero; ignored on reception.

Header Checksum: The 16-bit 1's complement of the 1's complement sum of all 16-bit words in the minimal forwarding header. For purposes of computing the checksum, the value of the checksum field is 0. The IP header and IP payload (after the minimal forwarding header) are not included in this checksum computation.

Original Destination Address: Copied from the destination field in the original IP header.

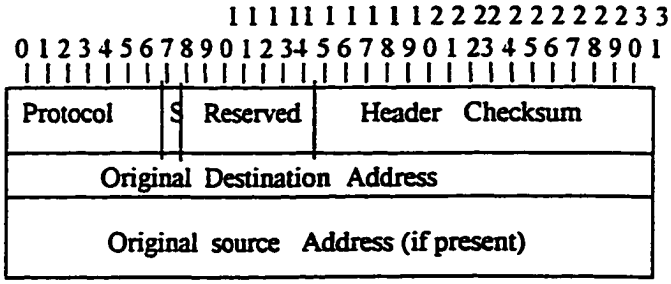


Figure 133:Minimal IP encapsulation format

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Original Source Address: Copied from the source address field in the original IP header. This field is present if the original source address present field is set.

Soft Tunnel State

One disadvantaged feature of ICMP error messages is that they are requited by the protocol to incorporate 8 bytes of the erroring datagram. Therefore when delivery of a datagram tunneled to a care-of address fails, the ICMP error returned to the home agent may not contain the IP address of the original source of the tunneled datagram.

Naturally, it makes sense for the home agent to try to notify the correspondent host (the source of the datagram which could not be delivered) in this situation. If the home agent keeps track of which datagrams have been tunneled to which care-of addresses (including the IP sequence number), the ICMP error return can be used by the home agent to indicate which datagram caused the error. If that determination is made, the ICMP error return can be *relayed* by the home agent to the correspondent node which sent the erroneous datagram.

When a correspondent node sends the datagram to the home network, and the datagram arrives at the home network, it seems inappropriate for the home agent to relay ICMP **network unreachable** messages without any change. In fact, from the viewpoint of the correspondent host, the tunnel should be invisible, almost as if it were an extension of the home link. So when the home agent can determine which correspondent host should receive the error, it makes great sense to convert the **network unreachable** message to a **host unreachable** message.

When a home agent is about to tunnel a datagram to a care-of address which just failed, it is quite feasible for the home agent to remember that the tunnel is broken. The home agent can then inform the correspondent host directly, using an ICMP **host unreachable** message. In fact, the home agent can keep track of other interesting tunnel parameters, especially including the path MTU for the tunnel, the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

necessary time to live (TTL) for encapsulated datagrams using that tunnel. This collection of tunnel parameters is called the soft-state of the tunnel. The IP-within-IP encapsulation specification, RFC 2003, recommends maintenance of soft state and gives specific rules for relaying ICMP messages.

Home Network Configurations

There are three basic configurations for home networks which are shown in figures 134 a, b and c. The first is a standard physical network connected by way of a router, with another node on the network acting as a home agent as in fig 134a. This configuration will be very popular, especially for enterprises starting to use mobile IP. If the home agent is also an enterprise router, the physical network layout can be conceptually simpler as illustrated in fig 134 b. In either case, wireless devices can be configured with IP addresses on existing physical networks with the help of bridging devices that cause the wireless packets to be bridged onto the physical network.

At the other extreme, it is possible to manage a home network that has no physical realization, called a virtual network, as in fig 134 c. The home agent appears as the router to the Internet, but when datagrams arrive at the network they are never forwarded. Instead the home agent encapsulates them and sends them to a known care-of address.

Proxy and Gratuitous ARP

In either configurations of the home network 134a or 134b, the home agent has to perform proxy ARP for the mobile node. Otherwise, existing Internet hosts on the home network would not be able to contact the mobile node after it has moved to some new care-of address.

In fact, hosts remaining on the home network which communicate with the mobile node while it is at home are likely to have ARP cache entries for the mobile node that become invalid when the mobile node moves away. For this reason, the home agent is required to broadcast *gratuitous* ARPs as soon as the mobile node moves away from the home network and registers a new care-of address. The gratuitous ARPs are supposed to have the effect of updating the ARP caches of each node physically attached to the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

home network so that now they resolve the IP home address of the mobile node into the link-layer address of the home agent. Similarly, when the mobile node returns to its home network, the node broadcasts gratuitous ARPs so that its home address is reset back to its link-layer address on the home network. Networks on which nodes are attached that do not work with gratuitous ARP should not be administered as home networks.

Because of the danger of creating irreparably outdated ARP caches, mobile nodes must never broadcast an ARP request or ARP reply packet on any visited network. If, for instance, a wireless mobile node were to broadcast an ARP request to find the link-layer address of the foreign agent that will broadcast the care-of addresses, any other wireless stations within the range could possibly create ARP cache entries for that mobile node. Those entries will be hard to wipe out after the mobile node moves away.

Datagram routing can be complex for datagrams going to the mobile node due to the *triangular routing*. There are optimization schemes that eliminate triangular routing problem through optimization methods though it necessitates changes in the correspondent nodes. Another problem facing mobile IP in today's Internet is that mobile nodes roaming in foreign enterprises look like an interloper and the firewalls and border routers administered at the foreign domain are usually configured to interrupt traffic to and from interloper nodes. One idea is to allow the mobile nodes to issue encapsulated datagrams using their care-of addresses as the outer source IP addresses. Note that using the care-of address as the source of IP address of the original datagram is a losing proposition as the correspondent node is keeping its track of sessions by way of the mobile node's home address, not the care-of address. Further research has been done in this area and many solutions are available today.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

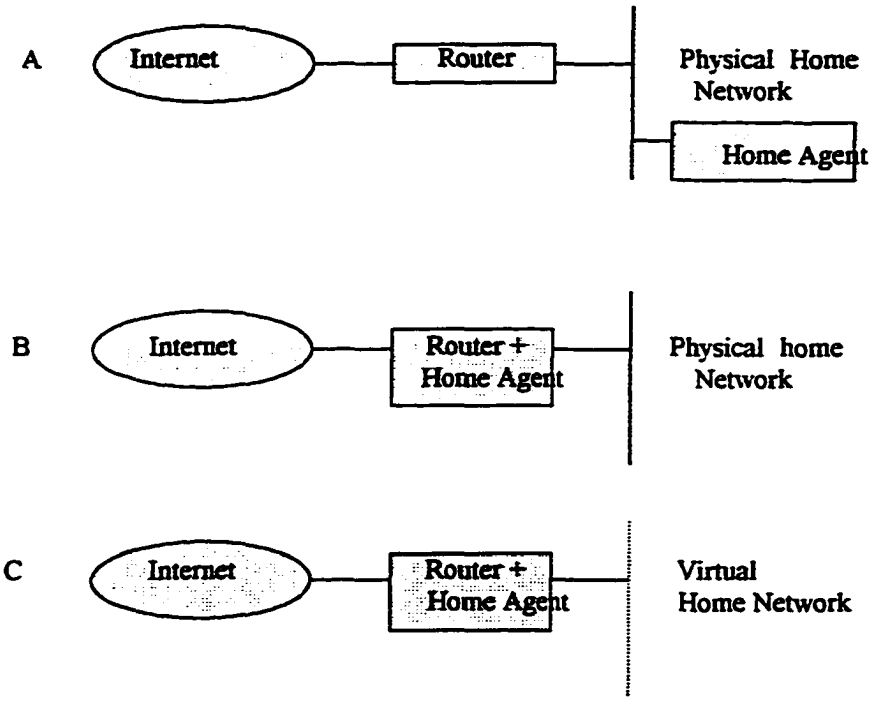


Figure 134:Various home network configurations

3.10 Internet and its architecture

Internet is a global collection of millions of heterogeneous networks, each running with its own distinct protocol, transmission technology and basic architecture, but all supporting the internetworkng (IP) protocol. The subnetworks are interconnected by IP switches and various transmission media. The international backbone is made of various intercontinental digital fiber optic cables as TPC-3, TPC-4, TPC-5, TPC-5NET (TPC is for Trans Pacific Cable), PacRimEast (between Australia and Hawaii), PacRimWest (between Asia and Australia), NPC, HAW etc. between Asia, Australia and USA, SEA-ME-WE-2, TAT-8, TAT-9, TAT-10, TAT-11, TAT-12, TAT-13 (TAT for Trans-American Transmission cable), TCS1 and many more between North America, South America and Europe etc. At the same time, the satellites as Telestar I, Telestar II, Telestar III series, Intelsat VI satellites have increased Megabits bandwidth by working at multi-Gegahertz bands. The global linkage of the fiber-optic cables are repeaterless, high-capacity and are complemented with SDH, thus facilitating network management and facility interworking. These numerous intercontinental cables and satellite channels provide fast, stable

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and disaster prone backbone for the internet. On top of that each country and continent has its own backbone connecting many networks where the backbone carries long distance traffic at multi-megabits speed. The leader in the national backbone technology is USA.

3.10.1 National Internet Backbone for USA

The very-high-speed Backbone Network Service (vBNS) is the result of the NSF's vision for the high-speed internet today. The original NSF model for the internet relied on a single hierarchy with the NSFnet (which became vBNS) backbone connecting regional networks, which provided connectivity to their own user communities. But the new model calls for multiple competing backbone service providers to serve their own user communities and for providers' networks to intersect at network access points (NAPs) for exchanging data and routing information. Running in parallel with the commercial backbones is the vBNS.

3.10.1.1 Role of the vBNS

The vBNS provides high-speed interconnection among the five Supercomputing Centers (SCCs), R&E institutions, and NSF-specified network access points (NAPs). This service provides high-speed data connectivity between R&E institutions and gives remote users access to high-performance SCC environments, supports the "metacenter" concepts among the SCCs, supports National Laboratory for Applied Network Research's web caching project, supports internet video conferencing and workgroup tools. In addition to offering high-performance network service, the vBNS also provides a testbed for new hardware, new protocols and new transmission systems in the networks. It was first implemented as wide-area IP/ATM network with OC-3 (155 Mb/s) infrastructure and now it is being currently upgraded to OC-12 (622 Mb/s) speeds.

3.10.1.2 Architecture Overview

The vBNS backbone interconnects five SCCs and four NAPs across USA where the SCCs are: Cornell Theory Center (CTC); National Center for Atmospheric Research (NCAR); National Center for Supercomputer Applications (NCSA); Pittsburg SuperComputer Center (PSC); San Diego SuperComputer

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Center (SDSC). The NAPs are: Sprint-New York, MFS - Washington D.C., Ameritech-Chicago. Pacific-Bell-San Francisco. Each SCC offers its router interface for three types of protocols: IP over FDDI ring; IP over high-performance parallel interface (HIPPI) and ATM UNI. Wide-area connectivity is offered by one or two OC-3 connections to the ATM wide-area network (WAN). Fig 135 shows the standard SCC equipment configuration. Cell level ATM interface (using video conference data etc.) is supported using FORE ASX-1000 whereas IP over HIPPI is supported by the Netstar Gigarouter and IP over FDDI is supported by CISCO 7507. Links to the R&E institutions are via a NAP or private interconnect at DS-3 (45 Mb/s) or OC-3 speeds. All these connections support IP and some support ATM as well.

MCI's commercial ATM network is being used as the backbone of the vBNS network. The commercial ATM network has an OC-3 backbone (about 15 OC-3 links), and offers OC-3 access for the five SCCs. The vBNS was the first "customer" on MCI's commercial ATM network and is the only customer with OC-3 access rates. The commercial ATM network carries vBNS traffic over a set of permanent virtual paths (PVPs). About 20 dedicated vBNS ATM switches (FORE ASX 1000s and LightStream 2020s) are collocated with MCI's commercial backbone switches. The vBNS PVPs traverse the shared backbone, but do not merge at the commercial backbone switches. PVPs run from one vBNS ATM switch to its neighbor and switching is supported using permanent virtual circuits (PVCs). Besides serving as switching points, also the vBNS ATM switches offer large buffers (multi-MB) to accommodate the high-bandwidth-delay product of high-speed TCP sessions. Use of the PVPs and dedicated switches allows the vBNS ATM switches to be viewed logically as a set of directly connected switches. The switches do not implement priority traffic class, traffic policing, traffic shaping and the switch queues implement FCFS discipline. In the shared backbone of the physical network, the vBNS is allocated a variable bit rate virtual path with a peak rate of 155 Mb/s and a sustained rate which is a fraction of the peak rate. This configuration is suited for bursty traffic.

3.10.1.3 IP over ATM

Switching IP traffic over the vBNS' dedicated PVPs is accomplished by two meshes PVCs.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

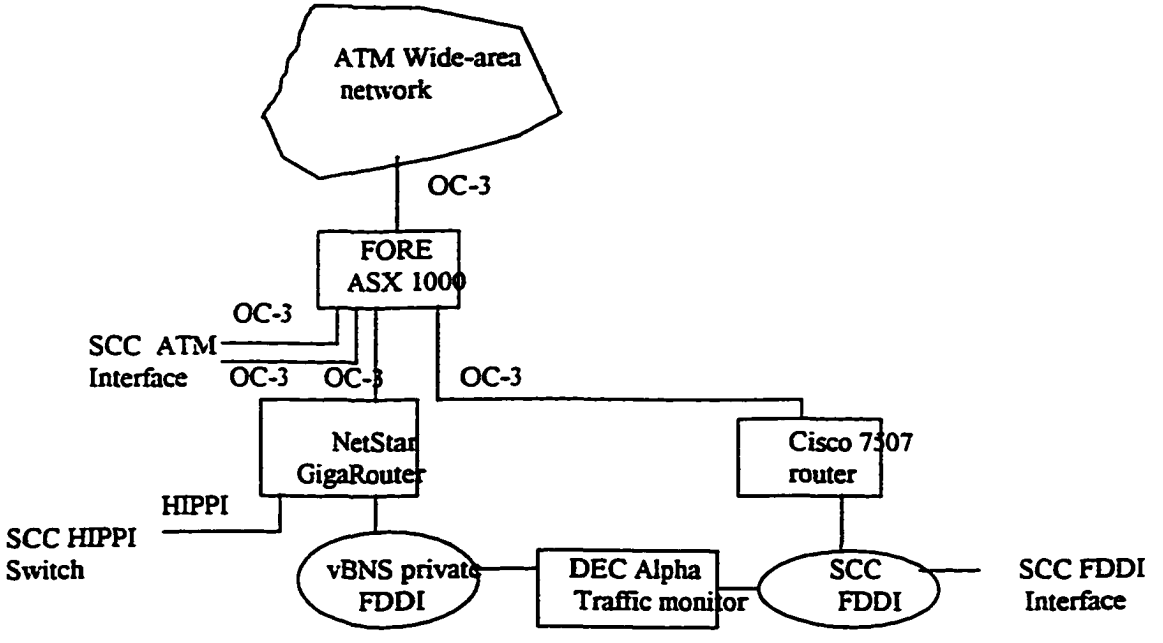


Figure 135:vBNS SCC architecture

One mesh of PVCs support "point-to-point" links between IP routers over which IP routing protocols are run; a second mesh supports a logical IP subnetwork (LIS) as well as non-IP ATM traffic. Fig 136 shows the vBNS backbone topology and fig 137 shows the logical connections between the ATM switches. The "point-to-point" PVCs provide a full mesh of interconnections between the vBNS ATM connected routers (which means all the routers on the network). Several routers are connected with a second "backup" PVC. This mesh of PVCs is utilized by the routers as if it were a mesh of point-to-point circuits. It is a flat architecture and each router is only one hop away from any other router: Open Shortest Path First (OSPF) is used as an interior gateway protocol (IGP) which enables the routers to share information about link costs and outages. Unless there is an outage, the lowest-cost PVC joining any two routers is used. In case of an outage, a "backup" PVC is used if available; otherwise, the information provided by OSPF is used to choose an alternate route. At the IP layer this alternate route will appear to be two or more hops long.

The LIS PVCs also provide a full mesh of interconnections between the vBNS ATM-connected routers. The mesh's endpoints also include additional ATM ports at the SCCs. Devices attached to these

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ATM ports can use IP to communicate with other members of the LIS, or use PVCs to exchange non-IP traffic (e.g. video over ATM) with other ATM-connected devices. Scheduled equipment upgrades will make it possible to replace the LIS PVC mesh with switched virtual circuits (SVCs). Using SVCs in stead of a mesh of PVCs will reduce the network complexity and increase network robustness.

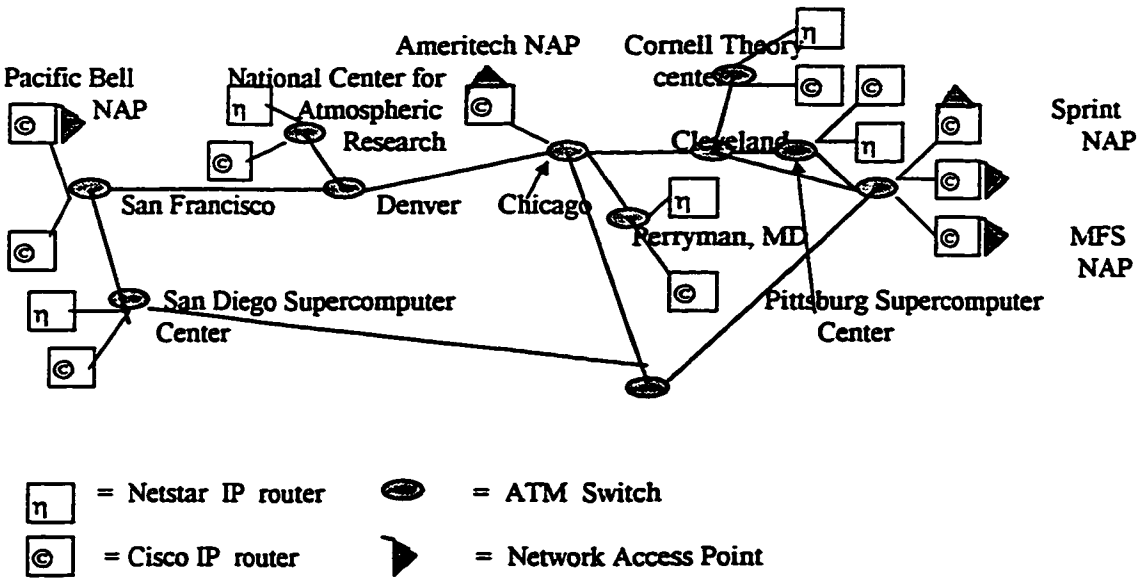


Figure 136: vBNS backbone topology

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

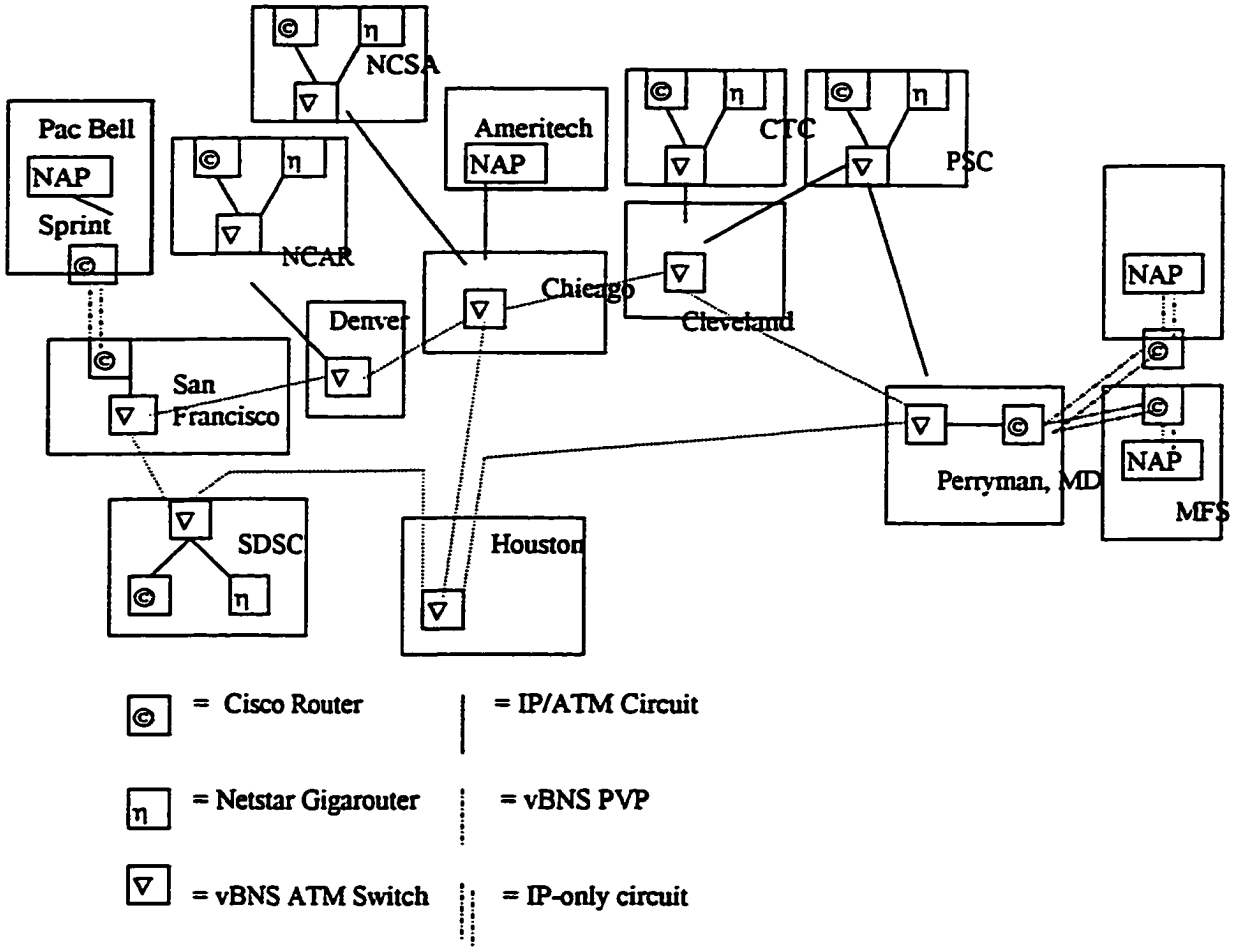


Figure 137:vBNS PVPs over MCI's commercial ATM network

3.10.1.4 BGP

BGP4 is used as the exterior gateway protocol (EGP). BGP peering sessions between vBNS routers and routers at the SCCs, R&E institutions, and networks that support R&E institutions are used to exchange routing information. Full Internet routing to vBNS backbone routers is supported via BGP peering with InternetMCI routers at the NAPs.

In order that the vBNS' resources are dedicated to and available for the community it serves, the NSF has established an Appropriate Use Policy (AUP) for the vBNS. The AUP is implemented on the vBNS using BGP communities. Two BGP communities have been set up: primary peers, which include the five SCCs and other NSF-specified R&E institutions; and secondary peers, which include other

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

networks that support R&E institutions such as Esnet (a Department of Energy network) and the National Aeronautics and Space Agency (NASA) Internet.

Primary peers are given all the routes the vBNS gets from both primary peers and secondary peers. Secondary peers are given only the routes the vBNS gets from primary peers. This scheme ensures connectivity between primary peers and between primary and secondary peers while preventing secondary peers from using vBNS for transit. Current vBNS primary peers include:

- The five SCCs: CTC, NCAR, NCSA, PSC and SDSC.
- University of Colorado at Boulder.
- University of Illinois at Chicago.

Current vBNS secondary peers include:

- Argonne National laboratory
- Canadian Network for the Advancement of Research, Industry and Education (CANARIE).
- Energy Sciences Network
- Interim Defense Research and Engineering Network (IDREN)
- NASA Internet

Performance measurements and traffic statistics for the vBNS are collected and analyzed regularly. Statistics and analysis results include utilization reports from router data gathered via SNMP, delay statistics and OC-3 link monitoring flow traces and also statistics gathered by MCI's own monitoring device. Round-trip between any two routing points in vBNS are shown as graph or tables, SNMP statistics can be reported in a graph produced according to the user's specifications or accessed via a network map of the vBNS showing the routers and switches at all locations.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.11 IPv6

Although IPv4 is universally accepted in the Internet and is very popular in general, many of its limitations have been taken care of in designing the new version called IPv6 or IPng on RFC 1752. After addressing performance, scalability, security, ease-of-configuration, and network management issues this new version really suits to be the protocol of the fast-evolving Internet.

- The network layer address in IPv4 is 32 bit long and very soon it will be saturated. In contrast IPv6 comes with 128 bit address space which can cover many more hosts, radio devices etc. on the Internet which seems to be the current of future.

- IPv6 paves the way for efficient multitiered routing hierarchies that limit the uncontrolled growth of backbone router tables. This has been a problem with IPv4 where the backbone routers tend to fill up the routing tables soon. IPv6 will allow Internet backbone designers to create a highly flexible and open-ended global routing hierarchy.

- IPv6 will also be meeting many application requirements. New IPv6 security features, for instance, give applications encryption and authentication services that are an integral part of the IP stack. For mobile business users and dynamic departmental staffs, the automatic configuration components of IPv6 will allow the efficient assignment of IP addresses without the delays and cost associated with manual address administration, which takes place in many current IP networks. IPv6's built-in quality-of-service features lay the groundwork for more deterministic end-to-end service levels in time-sensitive interactive and multimedia applications.

3.11.1 Addressing and Routing strategy in IPv6

At the level of the backbone where major enterprises and service providers come together, it will be better some kind of a hierarchical addressing system much like the long distance and international telephone system. Large central offices need only the three digit area code to route a call to another regional switch; but IPv4 uses somewhat haphazard style of address hierarchy to move traffic from one

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

subnet to another on the backbone of internet. Without an address hierarchy, the backbone routers will be required to keep addresses of each subnet of the world in its routing table which may become fairly large. With a hierarchy system, the backbone routers will need to use only a few prefixes of the address to decide how to route the traffic on the backbone.

Currently, IPv4 uses a method called Classless Inter-Domain Routing (CIDR), which makes flexible use of variable length network prefixes. With the flexible use, CIDR permits considerable route aggregation at various levels of Internet hierarchy, meaning backbone routers store a single entry in its routing table which provides reachability to many lower level networks. But much of the IPv4 address space was already formed before the current access provider hierarchy was developed. The lack of uniformity in current hierarchy creates complications in routing schemes. Also the Network Address Translator (NAT) was a workaround the problem of limited address spaces. Just because each host within an enterprise can not be given a unique internet address, internal addressing schemes were created and a NAT device would sit on the gateway to the internet translating the local address into a pool of global unique internet addresses that are passed on the backbone. But the NAT does not work well when the number grows or when the IP address is embedded within payload at higher level as in FTP, WINS registration services, Domain name services (DNS) etc.. Using CIDR style of address prefixes, IPv6 address space can be made to have route summarization for the backbone routers.

3.11.2 Autoconfiguration scheme

The "stateless" autoconfiguration scheme in IPv6 also supports automated configuration which does not require another extra manually configured server. Stateless autoconfiguration allows the host to configure its own addresses with the help of a local IPv6 router. Typically the station combines the 48-bit MAC address with a network prefix which it learns from its neighboring router and gives itself that address and notifies the internet router to be known by that address. Also autoconfiguration is an important enabler of mobile computing because it allows mobile computers to receive valid IP addresses automatically, no matter where they connect to the network.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.11.3 More streamlined format

The header structure has become much more simplified in IPv6. Many fields have been dropped, some have become optional and although the address has been increased to four times the old address length, but the header length (40 Bytes) has only doubled compared to the previous header. Additional processing efficiency is achieved by reducing the number of fields from 14 to 8 in the primary header. The payload length can be upto 64KB starting from 0.

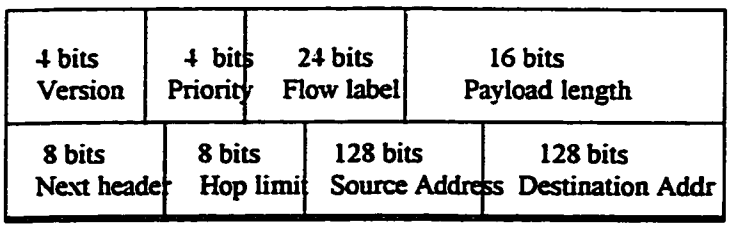


Figure 138:IPv6 header layout

The hop limit field is used by the routers to decrement the max hop value by 1 for each hop of the current end-to-end route. The hop-limit field is set to the appropriate value at the source node. If the value is decremented to zero, the datagram is discarded. This field will store a value upto 255 (meaning 255 hops) which is more than adequate for even the largest of networks in foreseeable future. Optional IPv6 header information is carried in "extension headers" following the fields in the header, but before the payload.IPv6 header extensions are now variable in length and have less stringent length limits. IPv6 gives network software designers a very straightforward technique for introducing new header options in the future. Option fields have already been defined to include explicit routing information created at the source node, facilitating authentication, encryption and fragmentation control.

3.11.4 Data security

Authentication, encryption and data security safeguards are standard requirements in enterprise networking today. IPv6 supports native data security options using header extensions. The authentication extension headers ensures that a packet is actually coming from the host indicated by its source address. This authentication is specially to safeguard against intruders who try to configure a host sending packets

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

with a forged source address. This type of source-address masquerading can spoof the server so that access can be gained to valuable data, passwords or network utilities. Because the authentication headers are defined in IETF standards, it is likely that various products from different vendors will achieve interoperable authentication services. IPv6 implementations are required to support MD5 authentication algorithm, though any algorithm may be used because the specification is algorithm-independent.

IPv6 also includes standard header extension that supports data encryption at the network layer. The encryption header extension provides fields to carry the encryption key and other handshaking information enabling interoperable encryption of the payloads in IP packets. IPv6 security headers can be used between hosts using a security gateway to emphasize an additional level of security with its own packet signing and encryption methods.

3.11.5 Multicast and anycast

One of the fastest growing business in internet is where a host transmits a stream of video, audio, news data to a group of related but physically dispersed endstations. Fig 139 illustrates the mechanism well where one set of packets are sent out, but they are delivered to a set of end-stations. A multicast-capable can automatically replicate the server's packets and route them to each subscriber in the multicast group using an efficient routing path. Routers use routing protocols as DVMRP (Distance Vector Multicast Routing Protocol) and MOSPF (Multicast Open Shortest Path First) to dynamically converge a packet distribution "tree" that connects all members of a group with the multicast server.

A new member becomes part of the multicast group by sending a "join" message to a nearby router. The distribution tree is then adjusted to include the new route also. Multicast services mean that server can send single packet which will be replicated and forwarded to the multicast group on an as-needed basis. This conserves both server and network resources and hence is superior to unicast or broadcast solutions. IPv6 broadens IP multicasting capabilities by defining a very large address space and a scope identifier that will limit the degree to which multicast routing information is propagated in the network.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

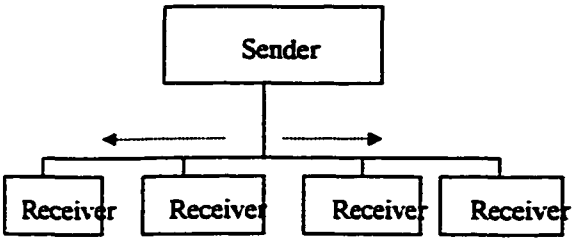


Figure 139:Multicasting to a few subscribers

3.11.6 Anycast Services:

Anycast services is an innovation of IPv6 which is conceptually a cross between unicast and multicast. Two or more interfaces on an arbitrary number of nodes may be designated as an anycast group. A packet addressed to an anycast group is delivered to only one interface, typically the nearest interface in the group according to the current routing protocol metrics. This is in contrast with multicast services which deliver packets to all members of the group. Nodes in an anycast group are specially configured to recognize an anycast address which are drawn from an anycast address space.

Because anycast is a new service, its applications have not been envisioned fully yet. Initially it is recommended that anycast services be limited to intermediate nodes. This would allow an enterprise to use a single anycast address for forwarding packets to a number of different routers on its ISP's backbone. If all of a provider's routers have the same anycast address, traffic from the enterprise will have redundant access points on the internet. If one router goes down, another in the group will receive the packets. The example may be more clear from the figure 140. As the functionality of anycast matures, it may become an important method for allowing end stations to efficiently access well-known services, mirrored databases, web sites, message servers etc. For example an enterprise may have a few Lotus Notes Servers all belonging to an anycast group. End applications of Notes will send the packets to the first available server through this method. This can become a cost-effective way of application load balancing for the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

servers. Anycast services could be used to provide redundancy in routing process by assigning all the DNS servers to the same anycast group.

3.11.7 Quality of Service

IPv6 has a 24 bit traffic flow identifier field in the header which can be used to implement quality of Service (QoS) network functions. IPv6 lays the foundation so that a wide range of Quality of Service (QoS) functions may be made available in a highly open and interoperable manner. In simple, this flow label can be used to identify some packets that need some special handling beyond the default best-effort forwarding. Flow-based routing could give internetworks some deterministic characteristics as with connection-oriented switching technology and telephony virtual circuits. For example, video or audio streams from a desktop could be given a flow label that tell routers that they need a controlled amount of end-to-end latency. Flow labels can also be used to provide traffic flow some specific level of security, propagation delay (e.g. in satellite transmission) or a fixed routing cost.

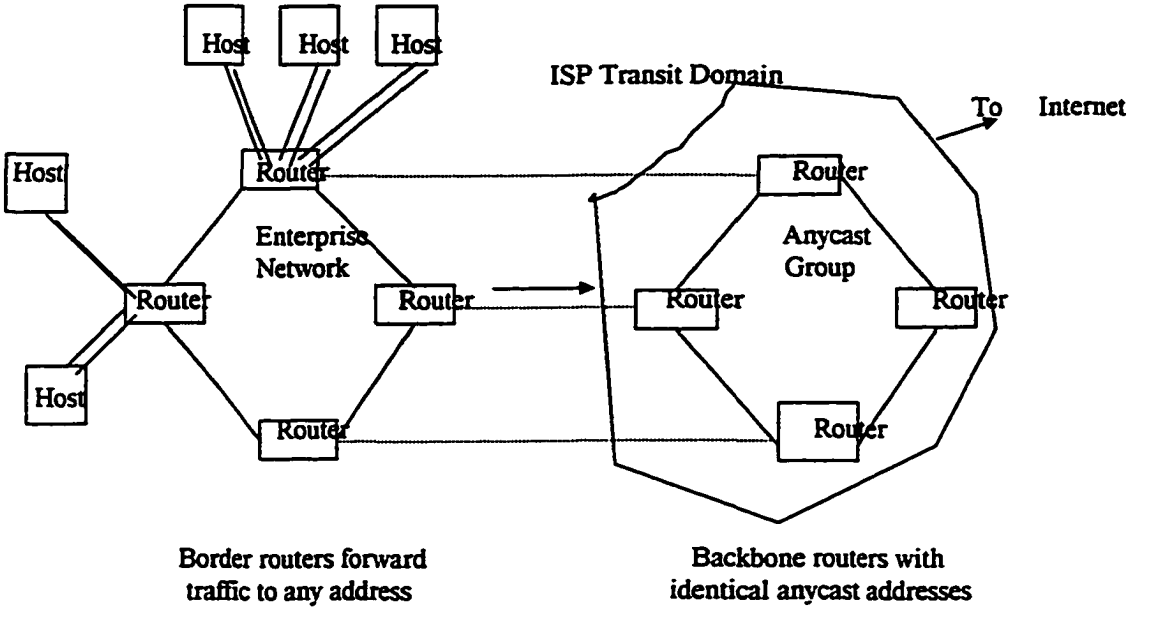


Figure 140:An example of Anycast

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.11.8 Transition to IPv6

For a smooth transition to IPv6,it is required to have downward compatibility for IPv4 and also accommodate IPv4 style of addressing. The solution from IETF has been dual-stack hosts-and-routers and tunneling IPv6 via IPv4.

3.11.8.1 Dual-stack transition method

Once a few nodes are upgraded to IPv6, they may need to communicate with IPv4 nodes and this is accomplished with the dual-stack IPv4/IPv6 approach. A great many hosts and routers in today's multivendor multiplatform network environment already support multiple protocol stacks. Many hosts run some combinations of IPv4, IPX, AppleTalk, SNA, DECnet, NETBIOS and other protocols. Addition of IPv6 will not be anything new. A host, when running dual IPv4/IPv6 stacks, will have access to both IPv4 and IPv6 resources. Routers running both protocols can forward traffic for both IPv4 and IPv6 endstations.

Dual-stack machines can use totally independent IPv4 and IPv6 addresses or they can be configured with an IPv6 address which is IPv4 compatible. Dual-stack nodes can use their conventional IPv4 autoconfiguration services (DHCP) in order to obtain their IPv4 IP addresses. IPv6 addresses can be manually configured in the 128 bit local host tables or obtained via IPv6 stateless or stateful autoconfiguration mechanisms, when available. It is expected that major servers will run in dual-stack mode indefinitely, or until all active nodes are converted to IPv6.

3.11.8.2 IPv6 DNS

Domain name service is something that administrators must consider before deploying IPv6 or dual-stack hosts. The current 32-bit name servers can not handle name resolution requests for IPv6 devices where the address is 128 bit long. In response to this issue, IETF designers have defined IPv6 DNS standards (RFC 1886). This specification creates a new record type called "AAAA" which will map domain names to an IPv6 address. Domain name lookups based on 128-bit addresses are also defined. Once an IPv6-capable DNS is in place, dual-stack hosts can interact interchangeably with IPv6 nodes. If

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

a dual-host queries a DNS and receives a 32-bit address, IPv4 is used; if 128-bit address is received, then IPv6 is used. On sites where DNS has not been upgraded to IPv6, hosts may resolve name-to-address mappings through the use of manually configured local name tables.

Applications that do not directly access the network stack do not need to be modified in order to run in the dual-stack environment. Network applications that directly interface with IP and related components will require updating if they need to run IPv6 protocol. For example, applications that access the DNS must be enhanced with the capability to request the new 128-bit records - a small change. Applications that make use of IPv6 security, quality of service and other features will need extensive updates.

3.11.8.3 Routing in IPv6/IPv4 networks

Routers running IPv6/IPv4 can be administered the same way as IPv4 routers can be administered today. IPv6 versions of the popular routing protocols such as Open Shortest Path First (OSPF) and Routing Information Protocol (RIP) are already under development. In many cases administrators will like to keep the IPv6 topology separate from IPv4 network, even though both may run on the same physical network-mainly for administration ease. In other cases it may be advantageous to align the two architectures by using the same domain boundaries, areas and subnet organization. Both approaches have their advantages.

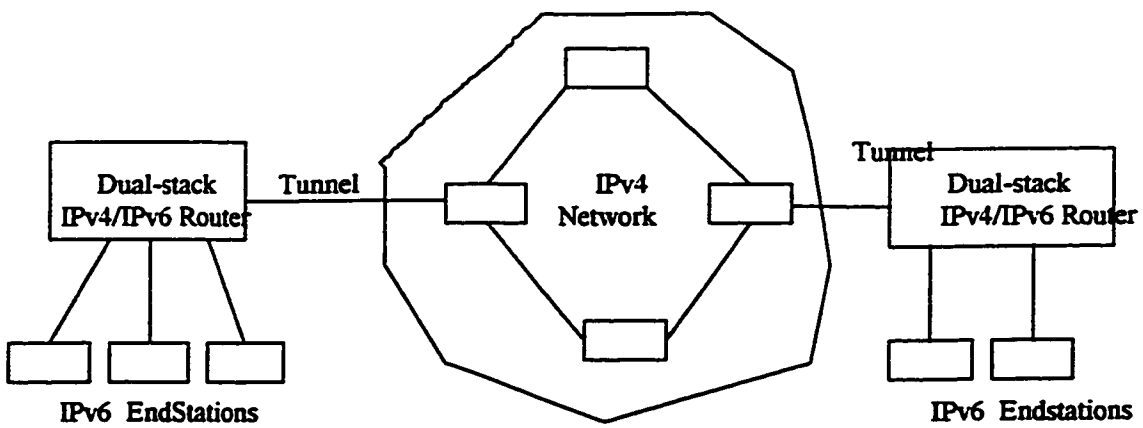


Figure 141:IPv6 over IPv4 Tunneling

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

An independent IPv6 architecture presents the opportunity to build a fresh hierarchical network address plan that greatly facilitate connection to one or more ISPs. This lays a foundation for efficient renumbering, route aggregation and the other goals of an advanced internetwork routing hierarchy.

In many organizations where IPv6 is implemented incrementally, there is a strong possibility that IPv6 hosts will not have direct connectivity to each other via IPv6 routers. IPv6 designers have fashioned transition mechanisms that allow IPv6 hosts to communicate over intervening IPv4 networks. This method is known as IPv6 over IPv4 tunneling, which encapsulates IPv6 packets in IPv4 packets.

3.11.8.3.1 IPv6 over IPv4 Tunneling

Tunneling allows early IPv6 implementations take advantage of existing IPv4 infrastructure without needing any change to the IPv4 components. A dual-stack router or a host at the edge of the IPv6 topology will simply append an IPv4 header to each IPv6 packet and send it as native IPv4 traffic through existing links. IPv4 routers will simply forward this traffic thinking this as IPv4 traffic. On the other end there is one dual-stack router or host that de-encapsulates the IPv6 packet and routes it to the final destinations using IPv6 protocols. To accommodate different administrative needs, IPv6 transition mechanisms include two types of tunneling: automatic tunneling and configured tunneling. To build configured tunnels, administrators manually define the IPv6-to-IPv4 address mappings at the two tunnel endpoints. On either side of the tunnel, traffic is forwarded with full 128-bit addresses. At the tunnel entrypoint, a router table entry is defined manually that dictates which IPv4 address to use to traverse the tunnel. This requires a certain amount of administration at the tunnel endpoints, but traffic is routed through the IPv4 networks dynamically, without the knowledge of the IPv4 routers. The 128-bit addresses do not have to align with the 32-bit addresses in any way.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Automatic Tunneling

Automatic tunnels use IPv4-compatible addresses which are hybrid IPv4/IPv6 addresses. Compatible addresses are created by padding extra zeroes before the IPv4 32-bit address to make it 128-bit long. When traffic is forwarded with compatible addresses, the device at the tunnel entry point can automatically address encapsulated traffic by simply converting the IPv4 compatible 128-bit address to a 32-bit IPv4 address. On the other side of the tunnel, the IPv4 header is removed to reveal the original IPv6 address. Automatic tunneling allows IPv6 hosts to dynamically exploit the IPv4 networks, but it does require the use of IPv4-compatible addresses which do not bring the benefit of 128-bit address space.

IPv6 nodes using IPv4-compatible addresses can not take advantage of extended address space of IPv6, but they exploit the other IPv6 enhancements, flow labels, encryption, authentication, multicast and anycast. Once a node is moved from IPv6 to IPv4-compatible addressing, there is always a chance to start using full 128 bit address space. IPv4-compatible addressing means that administrators can add IPv6 nodes while initially preserving their basic addressing and subnet architecture. Automatic tunnels are available when needed, but they may not be necessary in cases where major backbone routers are updated to IPv6 all at once. This can be achieved quickly and efficiently when backbone routers support remote configuration and update capabilities.

The combination of tunnels, compatible addresses and dual-stack nodes ensures that network administrators will have enormous flexibility and interoperability when they deploy IPv6.

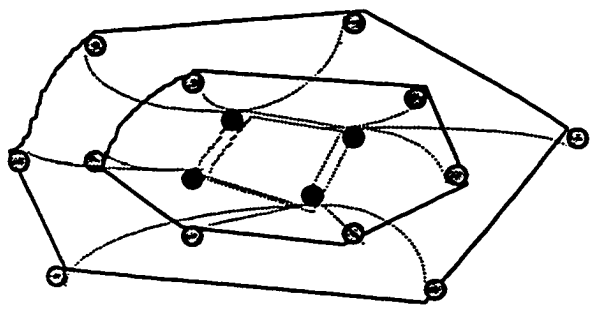
As a whole, with its extension headers and all other features as mentioned above IPv6 has incorporated all the features required by applications of advanced Gigabit communication and appears to be commonly used in the future.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4. Distributed Intelligent Information Network

4.1 Intelligent Networks(IN)

Intelligent networks is a positive step towards integration of networks and computers. As networks are faster, it is also perceived to become smarter and intelligent. Today, the users demand many facilities as call transfer, conference call etc. within their call. Some extra services are numerous and they are going to multiply further and further. Calling card facility, 800 dialing etc. need databases and thus database processing is also part of call processing. To support such various requirements of services during a call, or in the beginning of a call or at the termination of a call, a sound signaling protocol called CCS 7 is already being used in various switched networks and TCAP (Transaction Capabilities Applications Part) is already contained in CCS7. Using TCAP transactions can be fired as a part of signaling which is again part of call processing. Intelligent networks shows a clear path of how service logic or information processing can be combined in signaling. Intelligent networks is an extra network (which executes much logic) that runs in conjunction with any packet-switched or circuit-switched or a broadcast network and takes care of the information processing aspect for the served network and in fact many networks can share one intelligent network. An example topology is illustrated in fig 142. This With their information processing capabilities, they can be used in many different applications as user-network interface, network administration and control or in data handling.



Legends: ● Controlling computers ⋮ control path of IN Connection with the signaling network

| Transmission path of networks ● Switches of networks

Figure 142: A typical topology of two networks using an intelligent network

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Today, they are widely used in any public network to serve many of these functions. In very simple terms, these intelligent networks are made of some access points, some switches and processing nodes which have large computers. The computers have many functional components running with local and remote databases. These databases contain informations as calling-card related data, 800-number data etc.. There also may be more than one intelligent networks from different vendors providing various services which will be interconnected. There are various versions INs being used in USA, Europe, Japan and other countries as IN/1, IN/1+, IN/2 and AIN version 0 and version 1.

4.1.1 Evolution of IN

The history if IN is traced to predivestiture call waiting and business services such as Centrex facilities as offered by 1 ESS and then 5 ESS switches from AT&T about 25 years ago. These switches were part of the host network while possessing extra stored programs for processing logic etc.. During 1970s. Stored Program Control (SPC) technology was beginning to be introduced in systems whose functions were to support the network's management and maintenance. These systems were called Operations Systems (OS) needed to deal with the increasing complexity of the network and the services offered to its subscribers. Since then the functions provided in OSs have expanded to cover each aspect of network planning, engineering, administration and maintenance. Beginning in 1981, network intelligence reached a new plateau when AT&T introduced centralized databases to support Calling card and 800 service. These databases are located in Network Control Point (NCP) systems and are accessed by SPC switches via Common Channel Interoffice Signaling (CCIS) network.

After divestiture of the Bell System, the newly formed regional companies deployed centralized databases at Service Control Points (SCPs) [62] to support alternate billing service (ABS) and 800 calling. ABS provides calling card validation and other line information functions for collect and third-party billing. Access to SCPs from switches is provided via Signaling System 7 (SS 7) networks. The functionality in the switches, SS7 network, SCPs and OSs that support these services is collectively known

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

as IN/1. Soon there was a large effort to define the architecture of such a network at Bellcore and the new definition was called IN/2. This architecture included a greatly expanded set of switch, SCP capabilities known as functional components (FC), and a new system called the intelligent peripheral (IP). The FCs define the atomic functionality of the architecture and IPs provide service-assistance capabilities. Then a new design called IN/1+ evolved retaining SCP/SS7 based architecture, but including only those FCs which were service-independent. At the same time a forum called multi-vendor interactions (MVI) was organized by Bellcore encompassing Bellcore, the regional companies and the vendor community. The result of the MVI recommendation was included in the design of Advanced Intelligent Networks (AIN) which had two versions 0 and 1.

4.1.2 IN architecture

This architecture was mainly derived from a set of functional needs that came from the voice-band telecommunication services. To support the needs, a call model was setup that will include all the call states involved in a call and its architecture design is based mainly on this call model. In addition, IN architecture is highly flexible in terms of its physical realization and it can support other data networks.

4.1.2.1 Basic Components

Intelligent networks consist of sources and destinations of informations, network nodes, transmission facilities, service switching facilities and a host network. The source and destination databases

may refer to various applications as billing database, vendor databases, calling card database etc.. The network nodes have various functionalities as service logic processing, peripheral-related assistance, call routing etc. The transmission facilities are mostly digital and can be using ISDN link (144 kb/s), T1 link (1.544 Mb/s), OC-3 link using 45Nb/s or using DS-3 rates of data communication on 11-GHz digital radio channels. The IN core network has at least three nodes as the Service Switching Points (SSPs), Signal Transfer Points (STPs)[64] and Service Control Points (SCPs) connected by common channel signaling (CCS 7) links.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Service Switching Point (SSP)

These are the access points to the IN from the host network. Certain calls on the host network may trigger a service from the intelligent network while others may not. For example, an AT&T customer may only be eligible for a specific IN service or else the call is simply bypassed. If the call needs IN service, then a trigger fires in the SSP and transactions begin on the IN. The SSP is implemented as to combine call processing and service processing. The service processes generate TCAP primitives on CCS7 network. In generality, the SSP may be used in areas other than call control also. For example, in data networks also some service of the IN (e.g. some query) may be sought in which cases requests may be sent to the SSPs . In any case, as the service process runs in the SSPs, it may communicate with other processes in other SCPs using TCAP primitives. Actually speaking, it contains both functions as call control functions and service switching function. The call processing function provides the means for establishing the call services meaning call and connection handling. The service switching function generates the trigger for IN service, and interacts with call processing and service logic on behalf of these calls.

Signal Transfer Point

This modules act as the switches in the CCS7 network facilitating communications between SSPs and SCPs. The transfer takes place to dispatch the query to the right SCP via the CCS 7 links.

Service Control Point (SCP)

The SCPs are the information processors in the IN forming a distributed computing environment. It contains the service logic providing the logical control applied to a call and handles service related processing (e.g. analysis, translation, screening, routing etc.). They are the end points of the CCS network trying to find answers to queries from the various SSPs. The main hardware component in SSP is a powerful computer handling many SCP service functions in parallel in real time. The service management systems (SMS) serve the SCPs to which they are connected via the X.25 or BX.25 links. The architecture for an IN is shown in fig 143.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

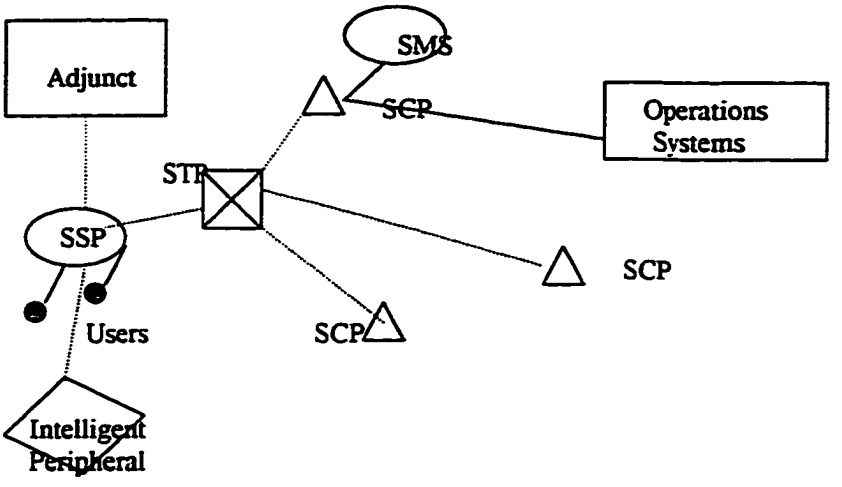


Figure 143:Architecture for AIN release 1.0

The functioning of SCP is better managed and streamlined by the Service Management System (SMS). It provides service provisioning, deployment and management control. Thus allowing access to all IN functional entities related to service logic and service data. The AIN services are provided through interactions between switching systems and the systems supporting AIN service logic.

One or more vehicles for providing AIN service logic are available in the network. Depending on implementation considerations and the services to be offered, service logic may be provided at a service control point (SCP), Adjunct system, services node.

Methods exist to allow network interaction with users. The interaction may involve the provision of service-specific announcements to a user or the collection of digits input by a user. The participant interaction resources may reside in a switching system or may be provided by an intelligent peripheral (IP). The switching system identifies conditions for service involvement, formulates requests for call processing instructions from AIN service logic, and responds to the instructions received. The call model provides a framework for formulating these requests and getting responses back from service logic.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Adjunct

The adjunct has a direct communication link to an SSP. Since a high-speed interface is envisioned between the adjunct and the SSP, there may be performance considerations that make the adjunct ideal for supporting services to give quick responses to the users (e.g. services as control provision of dial tone etc.).

IP

The intelligent peripheral (IP) is a system which control and manages resources such as voice synthesis, announcements, speech recognition and digit collection. The SSP routes a call to an IP as necessary to support the request of such resources by IN service logic. In establishing a call connection to an IP, the SSP also passes message parameters which instruct IP to perform specific user-interaction functions. When IP completes the requested functions, it returns any information collected from the user to the service logic in the SSP.

Operations Systems

AIN release 1 network operations provide memory administrations, surveillance, network testing, network traffic management and network data collection. The operations functions are located in the network components of the architecture, such as the switches and in operations applications (OA). OSs normally offer functions available in many network components.

4.1.3 International Standards

The CCITT group XI has provided Capability Set 1 (CS1) as a recommended set of advanced service enhancements for voice-band services on public networks. The basic objectives have been:

- Increase service velocity to enable the market-driven, rapid introduction of new services from conception to deployment.
- Broaden the range of services to go beyond the traditional voice and data service reaching a broader range as information services, broadband and multimedia bearer capabilities.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Enable a multivendor, competitive environment to ensure that the services will work correctly on several vendor's equipments.
- Evolve from currently existing networks.

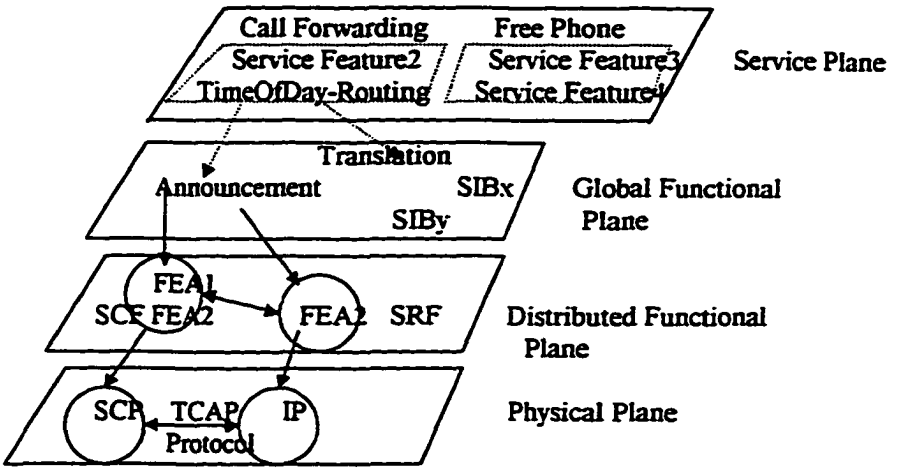
There are some type A services which are invoked during a call setup; type B services can be invoked at any point during the call.

4.1.4 Conceptual view of IN

To design an IN, the IN conceptual model is created first to serve as a modeling tool. This model has a four-plane representation which shows IN as an integrated framework. The service plane represents an exclusively service-oriented view of the IN, which does not contain any information regarding the implementation of the network's services. The global functional plane provides a view of the different functionalities of an IN network. The network is considered as a single entity, and this plane contain the call-processing model and service independent building blocks (SIBs). Service logic denotes all the service specific logic. The Distributed Functional Plane provides a view of the distributed functions of IN. This view is used to define the functional entities (FEs), their actions (FEAs), and their relationships. The physical plane models the physical aspects of the IN-structured network. It identifies the different types of physical entities, the functional entities they realize, and the protocols by which they communicate.

The IN framework, based on the above conceptual model, consists of a functional architecture, composed of functional entities and of a physical architecture composed of physical entities. Fig 144 shows the functionalities in different planes of abstraction starting from the physical plane up to Service plane.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Legend:

SIB: Service Independent Block	FEA: Feature Entity Action
SCF: Service Control Function	SRF: Special Resources Function
SCP: Service Control Point	IP: Intelligent Peripheral

Figure 144:IN conceptual model

The CCF (call Control function), SSF (service switching function), SCF (service control function), SMF (service management function) etc. have already been discussed above in terms of the physical entities. The SRF (special resource function) provide all the end-user interaction with the IN through control over DTMF receivers, voice recognition capabilities, protocol conversion, announcements and speech processing and it relates to the operation of IP. Service data function (SDF) handles the access to service-related and network data.

The call control agent function (CCAF) provides service access to the user and represents a user to call processing. The service data function handles access to service-related and network data. It provides a logical view of the data to SCF. The service creation environment function (SCEF) provides the capability to create, verify and test IN functions. The service management access function (SMAF) provides the interface to service management functions and includes checking authority to access functions. The set of CS1 service features are listed in table 4-1.Services are typically made of several features. In the global functional plane, a set of IN functionalities exist which regard the network as a

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

single entity. These building blocks are called service-independent building blocks (SIBs). They are global in nature, independent of physical architecture and a service feature is made of a few such SIBs. Each SIB has a unified and stable interface with one or more inputs and one or more outputs. The SIBs are realized in the Distributed Functional Plane (DFP) by functional entity actions (FEAs), which may reside in one or more functional entities (FEs).

Table 4-1: Set of CS1 service features

Reverse Charging	Call Distribution
Call Gapping	Call Limiter
Call Queuing	O_Call Screening
T_Call Screening	Closed User Group
Customer Profile Management	Follow-me Diversion
Origin Dependent Routing	Customized Recorded Announcement
Time Dependent Routing	User Prompter
Abbreviated Dialing	Authentication
Authorization Code	Off-Net Access
Off-Net Calling	Attendant
Mass Calling	Split Charging
Premium Charging	Private Numbering Plan
One Number	Customized Ringing
Call Logging	Call Forwarding in BY/DA
Call Forwarding	Personal Numbering
Call Waiting	Multi-way Calling
Meet-me Conference	Call Transfer
Call Hold with Announcement	Consultation Calling
Automatic callback	

The CS1 definition includes the following SIBs:

User Interaction - This SIB allows the exchange of information between a call party and the network.
Translate - This SIB determines an output information from some input information and additional features, based on various other input parameters. For example, a dialed number plus time of day and calling party identification may be translated to a destination directory number.
Log Call Information - This SIB records detailed information for each call into a file. The collected information may be used by SMS, but not by call-related services.
Limit - This SIB limits the number of calls that are allowed through an IN without causing congestion.
Charge - This SIB is used to determine special charging treatment for the call, where modified charging is required.
Queue -This SIB provides the sequencing of calls to be completed to a called party. This involves queuing incoming calls, dequeuing calls and timing the calls in the queue.
Basic Call Processing (BCP) - This SIB provides the SSF/CCF call processing capabilities.
Many more such SIBs also exist in the CS1 specification list [69].

The IN CS1 call model

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The call processing model is the set of states (Points in Call), transitions and detection points, which are used to illustrate the different states that a call can go through, from origination to termination. Normally a call model is separated in two parts: an originating call model and one terminating call model.

At present ANSI's T1S1 committee, ETSI from Europe are working with CCITT to come up with the international CS standards specification. At present, there are three sets of CS (CS1, CS2, CS3) standards already available.

With further time, more and more applications and vendor's networks will be incorporated into IN and one such application will be Information Network (INN).

4.2 Artificial Intelligence & Knowledge-Base

4.2.1 Introduction

Artificial Intelligence(AI) is the branch of computer science concerned with intelligent behavior which is based upon sound theoretical and applied principles. These principles include specialized data structures used in knowledge representation, algorithms to apply that knowledge and the languages and programming techniques in their implementation. Game-puzzles, automated reasoning, expert systems, natural language processing and semantic modeling, machine learning are some application areas of AI. These various aspects are widely used today in many areas of networks as cost analysis in routing, traffic analysis, congestion detection, distributed applications etc.

One important area of AI's application is Expert Systems which works on a particular domain of problem. Many expert problem solvers have been created in the past (e.g. DENDRAL, MYCIN, SHRDLU, PROSPECTOR etc.) for last 25 years. A sophisticated expert search engine is given a problem to solve and it searches a universe of facts and rules to find some solution to the problem. The techniques

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

used for searching are very sophisticated, various and varies with the problem-domain. Some key invariant knowledge about the problem-domain is built into the engine and the rest of the knowledge is maintained as a knowledge-base. Normally an Expert System comes as a shell containing formalism for representing knowledge base, a development environment and a run-time environment.

During the development, more and more knowledge is added into the system by the knowledge engineer. This unstructured knowledge is transformed into manageable structures through analysis. The knowledge can be structured in various levels of abstraction as shown in fig 145.

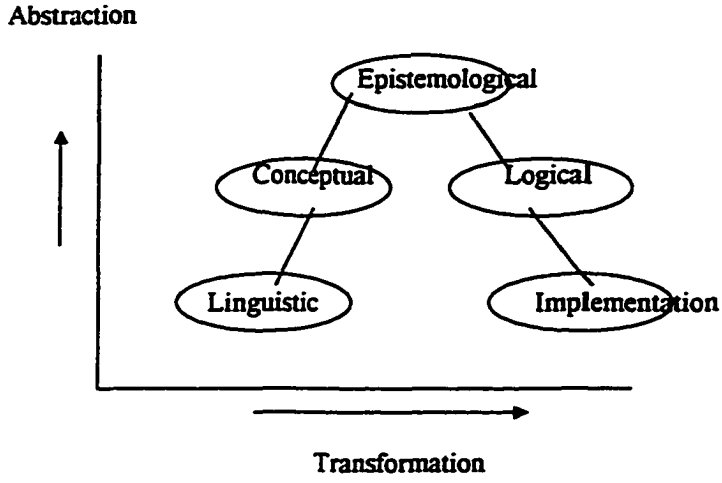


Figure 145:Levels of abstract in knowledge analysis

The epistemological knowledge is the highest level of abstraction that can be achieved while analysing knowledge and is most general view of knowledge.

4.2.2 Knowledge-Base Systems

It is basically a collection of simple facts, general rules and structures. The integrity rules are more explicitly stated than implicitly as in database systems. Knowledge is a symbolic representation of some aspects of the universe. Knowledge base systems are distinct from conventional database systems in four ways:

- Knowledge bases typically contain explicitly represented rules as well as simple facts.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Knowledge base storage structures typically have low structural semantic content compared with database structures.
- Knowledge bases systems include components for the automatic maintenance of semantic integrity in addition to components for syntactic checking.
- Knowledge base systems include components which can make inferences over the knowledge base, thus providing a deductive retrieval facility.

Because knowledge-bases are used in database technology, formal logic system, expert systems and natural language processing, these different areas share a notion of the view of the universe which may be defined as 'a set of concepts and rules that can be used to formulate and rationalize one's perception of parts of the universe'. Such views help one to identify, name, classify and represent those components. In databases, this view is given by the data model made of entities, attributes, relationships and integrity rules etc. In formal logic, the components of the universe is 'mathematical' structures. For example, in first order predicate logic parts of the universe are regarded as first order relational structures each consisting of a set of first order relations and functions defined on this set. Fig 146 illustrates with an example how two knowledge representations may look about a set of people consisting of John, Jim and Mary where John is married to Mary.

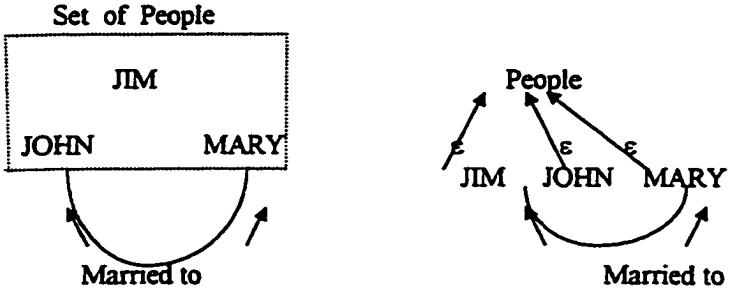


Figure 146:Various representations of the same knowledge

Knowledge-base representations can be many ranging from natural language (as in natural language applications), records, links and tables (as in databases), formal logics as propositional logic or first order

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

predicate logic etc., production rules, 'slot and filler'[82] representations. There are even more as functional representation (using entities and functions) etc.. 'Slot and filler' representation allows representing structures within the universe showing how the concepts are organized in the universe. 'Slot and filler' is a generic representation and its formalisms include 'frames', 'nets', 'conceptual dependency graph' and scripts. Slot and filler approach facilitate pattern recognition, inference of generic properties, handling of default values and detection of errors and omissions in a body of knowledge. Some important formalisms as formal logic and slot and filler approaches may be viewed in broader details.

4.2.2.1 Formal Logic

There are many branches in formal logic each of which has a well defined language for the representation of knowledge and also well defined methods for reasoning using consistency verification or deduction etc.. Assertions about parts of the universe may be formulated in a precise and unambiguous way by use of formal languages. For example, consider the following formula in the language of formal propositional logic:

(Pat is a boy) $\vee$ (Pat is a girl)

Well formed formulas of a formal language can be constructed by combining simple components called 'atomic formulas' (or propositions) using logical connectives as:

$\wedge$ - and $\vee$ - or $\neg$ - not $\rightarrow$ - implies

The atomic formulas always carry a truth value either as a constant or through an assignment.

The semantics of a formal language determine how meaning may be ascribed to atomic formulas, and how this meaning can be extended to give meaning to wffs in which the atomic formulas occur. The semantic effect on a well formed formula is determined from the truth assignments of the atomic formulas.

Truth assignment

This is a function which assigns truth values to the individual atomic formulas. A truth assignment is appropriate if it assigns truth values to each individual atomic formula. For example, consider a wff as

(Pat is a boy) $\rightarrow \neg$ (Pat is a girl). An appropriate truth assignment is :

α= { (Pat is a boy) := true,
(Pat is a girl) := false }

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Any truth assignment, ∂, can be extended to a function $\partial^{\cdot}$ which includes the following assignments and also the ones above:

(1) (Pat is a boy) $\wedge \neg$ (Pat is a girl) := true,
(2) (Pat is a boy) $\rightarrow \neg$(Pat is a girl) := true,
(3) $\neg$(Pat is a boy) :=false, etc.

Satisfiability

Let ∂ be some truth assignment and let ∂' be the function obtained by extending ∂. For any formula F for which ∂ is appropriate, if ∂' (F) is true, then we say variously, that ∂ 'satisfies' or 'verifies' F or that F is 'valid' in ∂. For example, α above satisfies the formula (2) above.

A formula or set of formulas is satisfiable if there is at least one truth assignment that satisfies it. The examples (1), (2), (3) are all satisfiable formulas. The following is an unsatisfiable formula:

(Pat is a boy) $\wedge \neg$ (Pat is a boy).

Universal validity

One formula is universally valid if every truth assignment which is appropriate will satisfy it. The following is a universally valid formula:

(Pat is a boy) $\vee \neg$(Pat is a boy). The universally valid formulas in propositional logic are called tautologies.

Equivalence

Two formulas are equivalent if they are assigned the same truth value for each truth assignment which is appropriate to both and this is expressed by logical connective $\leftrightarrow$. The following is an example of equivalence:

$[A \rightarrow B] \leftrightarrow [\neg A \vee B]$ where A and B are wffs.

Logical Consequence

A formula F is said to be a logical consequence of a formula or a set of formulas S, if F is satisfied by all truth assignments which satisfy S. Logical consequence is denoted by:

$S \Rightarrow F$. The following example is a logical consequence:

$\{A \vee B, \neg A\} \Rightarrow B$ Or one can say that F follows from S.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.2.2.1.1 Formal Deduction Systems

Given a set of wffs, S, it is possible to derive or deduce logical consequences of S by methods other than those described above. In particular, it is possible to deduce logical consequences of S by syntactic operations only. This means by structurally manipulating S without going into truth values, it is possible to derive new formulas which will be logical consequences of S and this method is called formal deduction system. There are various types of formal deduction systems and the type to be described here is called 'axiom systems'.

An axiom system consists of a formal language such as those described already, a set of inference rules, and a set of logical axiom schemas.

Inference rules

The inference rules define the syntactic operations by which new formulas can be generated from given formulas. Examples are the following in which P and Q are wffs:

R1: From $P \rightarrow Q$ and P, infer Q. This is called modus ponens.

R2: If F is a universally valid formula, then so is F' where F' is obtained from F by the consistent substitution of any subformula in F by any other sub-formula. For example, if $B \vee \neg B$ is a universally valid formula, then so is $A \vee \neg A$. This is the substitution rule.

Logical Axioms

The logical axioms comprise a set of templates for some of the universally valid wffs of the formal language. The set of logical axioms of an axiom system is generally chosen such that all universally valid formulas of the language can be generated from it using the inference rules of the axiom system.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Formal proofs

The derivation of a formula F from a set of formulas S using an axiom system A, is called a formal deduction or 'formal proof' of F from S in A. A formal proof of F in A is a finite sequence F1, F2, ...,Fn, of formulas such that F = Fn and for each i, $(1 \leq i \leq n)$:

either (a) Fi is a logical axiom of A

or (b) $Fi \in S$

or (c)Fi is generated from previous formulas of the sequence according to one or more of the inference rules of A

where $\in$ stands for 'member of the set'.

The existence of a formal proof of F from S in A, is denoted by:

$S \Rightarrow_A F$

Example

As example of a trivial formal proof, consider the following:

(a) axiom system = [logical axioms: $\{ P \vee \neg P,\ \neg\neg P \rightarrow P,\ [P \wedge Q] \rightarrow P,\ [P \wedge Q] \rightarrow Q \}$; inference rules: R1 and R2 from above]

(b) S = {(Pat is a student) $\wedge$ (Pat is a girl), (Pat is a girl) $\rightarrow$ (Pat is a human) }

(c) The formula to be proved is (Pat is human).

The proof proceeds in three stages:

1. Using the 4th logical axiom and the inference rule R2 one can infer the following formula:
   [(Pat is a student) $\wedge$ (Pat is a girl)] $\rightarrow$ (Pat is a girl)
2. From the formula derived in step 1, the first formula in S, and the inference rule R1, one can infer the following formula: (Pat is a girl)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3. Again from the formula derived in step 2, the second formula in S, and the inference rule R1, one can infer the following formula:

   (Pat is human).

The proof is now complete.

Commonly used inference rules

Modus ponens and substitution are only two of several rules of inference which are used in everyday reasoning. Other rules include the following, where P and Q stand for any wff:

R3: conjunction

from P and Q, infer P∧Q.

R4: Simplification

from P ∧ Q, infer P.

R5: Addition

from P, infer P ∨ Q.

R6: Transposition

from P→ Q, infer ¬Q → ¬P

R7: Hypothetical syllogism

from P→Q and Q→R, infer P→R.

R8: disjunctive syllogism

from P ∨ Q and ¬P, infer Q.

Axiom systems vary in the choice of inference rules and logical axioms. Some inference rules are closely related to particular logical axioms. For example, the rule R4 and the logical axiom [P ∧ Q] → P are 'equivalent' in the sense that the logical axiom can be used to infer P from P∧Q when used in conjunction with the rule modus ponens.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Axiom systems are not the only type of formal deduction system. Other systems have been built which contain only inference rules and no logical axioms.

4.2.2.1.2 Soundness and Completeness

Soundness

A formal deduction system A is said to be 'sound' if all of the formulas that can be derived from any set of formulas S using A. are also logical consequences of S. Soundness is denoted by:

$$S \Rightarrow_A F \text{ implies } S \vdash_A F$$

This means in a sound system, if a formula F can be derived from a set of formulas S, then one can be certain that F is a logical consequence of S. Soundness is an essential property of any useful formal deduction system.

Completeness

If for some formal deduction system A, every logical consequence of any set of formulas S can be derived from S using A, then A is said to be complete. Completeness is denoted by:

$$S \vdash_A F \quad \text{implies} \quad S \Rightarrow_A F$$

Completeness is a desirable though not an essential property of a formal deduction system.

It can be seen that the inference rules of a sound system must be closely related to the semantic definitions of the logical connectives of the language of that system. However, it should be noted that although inference rules are themselves obtained using some argument involving semantics, their use in formal proofs involves purely syntactic operations.

4.2.2.1.3 Theories and Theorem Proving

Theories

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A theory consists of a formal system together with a set of wffs which are known to be true in some set of intended interpretations. These wffs are called the 'proper axioms' of the theory. The following is an example of a set of proper axioms:

S = { P1: (Ali is a man) $\vee$ (Ali is a woman)

P2: $\neg$ [(Ali is a man) $\wedge$ (Ali is a woman)]

P3: (Pat is a man) $\vee$ (Pat is a woman)

P4: $\neg$ [(Pat is a man) $\wedge$ (Pat is a woman)]

P5: [(Ali is married to Pat) $\wedge$ (Ali is a man)] $\rightarrow$ (Pat is a woman)

P6: [(Ali is married to Pat) $\wedge$ (Ali is a woman)] $\rightarrow$ (Pat is a man)}

The intended interpretations of a theory are those truth assignments which satisfy all proper axioms of the theory according to the rules of the formal system. For example, the following are some of the intended interpretations of the theory T' which consists of a sound and complete axiomatisation of propositional logic together with the set S of proper axioms given above.

$\partial 1$ = { (Ali is a man) := true, (Ali is a woman) := false, (Pat is a man) := true, (Pat is a woman) := false,

(Ali is married to Pat := false}

$\partial 2$ = {(Ali is a man) := false, (Ali is a woman) := true, (Pat is a man) := false, (Pat is a woman) := true,

(Ali is married to Pat) := false }

$\partial 3$ = { (Ali is a man) := true, (Ali is a woman) := false, (Pat is a man) := false, (Pat is a woman) := true,

(Ali is married to Pat) := true }

... etc.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The theory T holds for situations in which 'Ali is a man or a woman but not both', 'Pat is man or a woman but not both', and in which 'if Ali is married to Pat and Ali is a man then Pat must be a woman', and 'if Ali is married to Pat and Ali is a woman then Pat must be a man'.

Theorem-proving

A 'theorem' of a theory is a formula which has a formal proof in that theory. The fact that F is a theorem of a theory T is denoted by:

$T \Rightarrow F$

For example, suppose that we have a theory T'' which consists of the proper axioms S above together with the following axiom system:

logical axioms = { L1:$[P \rightarrow Q] \rightarrow [\neg P \vee Q]$,

L2: $\neg[P \wedge Q] \rightarrow [Q \rightarrow \neg P]$,

L3: $[P \wedge Q] \rightarrow [Q \wedge P]$,

L4: $[P \vee Q] \vee R \rightarrow [P \vee Q \vee R]$,

L5: $\neg[P \wedge Q] \rightarrow [\neg P \vee \neg Q]$}

Inference rules = R1, modus ponens

R2, substitution

R7, hypothetical syllogism

The following is a theorem of the theory T'':

[¬(Ali is married to Pat) ∨ ¬(Ali is a man) ∨ ¬(Pat is a man)]

The proof proceeds as follows:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

step 1. From P4, L2, R1, R2 infer the following:

F1: [(Pat is a woman) → ¬ (Pat is a man)]

step 2: From P5, F1, R1, R7 infer the following:

F2:[[(Ali is married to Pat) ∧ (Ali is a man)] → ¬(Pat is a man)]

step 3: From F2, L1, R1, R2 infer the following:

F3:[¬[(Ali is married to Pat) ∧ (Ali is a man)] ∨ ¬ (Pat is a man)

step 4: From F3, L5, R1, R2 infer the following:

F4:[[¬(Ali is married to Pat) ∨ ¬(Ali is a man)] ∨ ¬(Pat is a man)]

step 5: From F4, L4, R1, R2 infer the following:

F5: [¬(Ali is married to Pat) ∨ ¬ (Ali is a man) ∨ ¬ (Pat is a man)]

Theorems of formal deduction systems

The formulas which can be derived within a formal deduction system alone with no proper axioms are theorems of that deduction system itself. The theorems of a complete and sound deduction system are all of the universally valid formulas of the language of that deduction system.

Consistency

A theory T is 'inconsistent' if and only if (iff):

$T \Rightarrow F$ and $T \Rightarrow \neg F$ for some formula F

Theory T is consistent iff it is not inconsistent.

4.2.2.1.4 Purely Syntactic Theorem-Proving

Given some theory T, one can prove whether or not a formula F is a theorem of T by repeatedly using the inference rules of the formal deduction system and deriving new formulas. This approach is

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

regarded as using syntactic reasoning since it involves only the structural manipulation of formulas. The consistency of a theory T may also be determined in this way in some cases by generating *all* theorems of T and checking to see if any formula F and its negation ¬F are derived. in which case T is inconsistent.

4.2.2.1.5 Purely Semantic Theorem-Proving

One can sometimes determine whether or not a formula F is a theorem of a theory T in a sound and complete deduction system by constructing a truth table to see if F is valid in all satisfying truth assignments of T. For example, consider a trivial theory T''' which has the following proper axioms:

¬P where P might stand for (Pat is a woman)

P ∨ Q where Q might stand for (Pat is a man)

One can prove that Q is a theorem of T''', in a sound and complete axiomatisation of propositional logic by constructing a truth table with assigned truth values. If Q is true (which it is) in all cases in which ¬P and P ∨ Q are true. In theories of sound and complete deduction systems, the syntactic and semantic approaches to reasoning are equivalent.

4.2.2.1.6 Refutation Procedures

Instead of applying purely syntactic or purely semantic reasoning, one can apply syntactic procedures followed by semantic checks in order to determine whether a formula F is a theorem of a theory T or not. One such approach may involve the following steps:

1. Add the negation of F to the set of proper axioms S of the theory T to obtain a new set of formulas S' = S ∪ ¬ F, where ∪ is the set union operator.
2. Repeatedly apply inference rules to S' to derive new formulas which are added to S'.
3. Test S' for satisfiability.
4. If S' and therefore S ∪ ¬F is unsatisfiable, then F must be a theorem of T.

Such procedure in which the formula being tested is negated are called 'refutation' procedures.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.2.2.1.7 Decidability

So far, examples have been given of theorem-proving and consistency checking which are 'decidable'. This means, it was preassumed that it is at least theoretically possible to determine whether or not a formula is a theorem of some theory or a theory is consistent. But it is not possible to determine in all cases. Purely syntactic methods are guaranteed to terminate in finite time if there are only finite number of deductions that are possible, or in the event that the formula searched for is indeed a theorem. Purely semantic methods are guaranteed to terminate in finite time if there are only finite number of truth assignments or the formula is not a logical consequence of the theory.

Two particular logics are of main interest to the researchers: propositional logic and first order predicate logic.

In classical propositional logic, well-formed formulas (wffs) are constructed from atomic formulas and the logical connectives $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$. There are automated theorem proving using resolution principles, automated reasoning etc. in this logic system.

But in propositional logic, one is allowed to reason about propositions. However, there is no way to reason about the relationships that exist between the individual entities in the universe of discourse. In contrast, predicate logic is more expressive in power. It can describe and reason about parts of the universe viewed as 'relational structures', where a relational structure consists of a set of entities E, a set of relations defined on E and a set of functions defined on E. Entities in a sub-set of E are known as 'distinguished entities'. For example, there can be a predicate relation as Aged (Bill, 30). A formal language of first order predicate logic can support predicate symbols, individual variables, constants, functions, all logical connectives as in propositional logic, two quantifiers as $\forall$ and $\exists$ which stand for 'for all' and 'there exists one' and brackets. Similar things as interpretations, valuations, semantic denotations, models, theories, inference rules etc. are defined in predicate logic. Automated search procedures, automated theorem proving etc. can be done more powerfully in first order predicate logic.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

All the basic concepts of entities, relations, attributes, integrity rules, relational calculus in relational databases are based on the first order predicate logic.

There are also some limitations on the power of predicate logic: it is not expressive enough; it can not handle incomplete, uncertain, vague knowledge; the algorithms for manipulating knowledge which are derived from logic are inefficient. This has led to work in other parts of logic as many-sorted logic, situational logic, non-monotonic logic, many-valued logic, fuzzy logic, modal logic, temporal logic, epistemic logic, higher-order logic and intensional logic. Non-monotonic logic deals with incomplete knowledge where addition of an assertion to a theory can invalidate many conclusions which were previously made. Fuzzy logic is more of a 'commonsense' logic which falls outside conventional logic and includes uncertainty measures. Test-score semantics is used to assign a meaning to a proposition in fuzzy logic. In test-score semantics, the meaning of a proposition is given by the procedure which is used to compute the test score of that proposition. It deals with incomplete knowledge as 'usually snow is white'.

Modal logic is a logic that allows one to reason with statements which are in subjunctive moods rather than in indicative mood. These subjunctive statements assert what must be, ought to be, might be, is believed to be, hoped to be, will be in the future etc.. These statements are distinct from indicative statements which simply assert what is. Classical truth-functional logics are concerned with indicative statements. Modal statements can be detected by the presence of modal operators. For example, consider the following statements:

1. John has appendicitis.
2. It is the case that John has Appendicitis.
3. It is possible that John has appendicitis.

Statements 1 and 2 are not modal. Statement 2 is true iff statement 1 is true. But statement 3 is modal. It is true if 1 is true but may be interpreted as true or false if 1 is false. Statements with modal logic normally contain operators as 'it is possible that', 'it is necessary that', 'it will always be the case', 'it is obligatory that', 'it is permissible that' ,'it is known that', 'it is believed that' etc.. Modal logic is concerned with states of affairs or possible worlds in addition to the one that exists.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Temporal logic, is concerned with time-varying relational structures. Various approaches have been

developed to deal with variations in time: extensions to classical logics, various types of modal logic and some forms based on intensional logic. Classical first order predicate logic can be used to reason about time by regarding 'time points' as being like all other entities in the domain of a relational structure. The time points are related to each other by predicates in the same way that other entities are related. For example, one may define two predicates 'et' and 'ss', such that et (t1, t2) means time t1 is earlier than t2 and ss (t1, t2) means that time t2 is immediate successor of t1. Using these predicates, one can write wffs as

$\forall x \forall y [et(x, y) \rightarrow tpt(x) \wedge tpt(y)]$ where tpt(x) means x is a time point.

or

$\forall x \forall y [et(x, y) \rightarrow \neg et(y, x)]$

For example, to indicate an entity's age never decreases, one will have a wff as

$\forall v \forall w \forall x \forall y \forall z [age(x, y, z) \wedge age(x, w, v) \wedge et(z, v)] \rightarrow \neg$less than (w, y)

Epistemic logic is used to reason about knowledge possessed by several reasoning agents in stead of one reasoning agent. In the logics considered so far, one is concerned with only one reasoning agent. But in this logic, one describes how the reasoning goes about the knowledge possessed by several agents. By knowing the limit of the other agents' knowledge and reasoning ability, it is possible to infer new knowledge. Much of the discussion is derived from Konolige (1982).

One begins with description of Konolige's logic.

The logic

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The logic which Konolige defined is called KI4, and is based on a modal logic called K4. K4 includes a set of indexed modal operators:

$\{ \boxed{s} \mid s \in \text{a set of agents}\}$

where the formula $\boxed{s}\ \alpha$ means that agent s knows α. The agent 0 is a fictitious agent called 'FOOL', whose knowledge is common knowledge.

The Logical axiom schemes of K4

The logical axiom schemas of K4 are:

A1: all propositional tautologies

A2: $\boxed{s}\ \alpha \rightarrow \boxed{s}\ \boxed{s}\ \alpha$

A3: $\boxed{s}\ [\alpha \rightarrow \beta] \rightarrow [\boxed{s}\ \alpha \rightarrow \boxed{s}\ \beta]$

A4: $\boxed{s}\ [\alpha \rightarrow \beta] \rightarrow [\boxed{s}\ \alpha \rightarrow \boxed{s}\ \beta]$

A5: $\boxed{0}\ [\alpha \rightarrow \boxed{0}\ \boxed{s}\ \alpha]$

where α and β denote arbitrary formulas, and s denotes an arbitrary agent.

A5 is the 'common knowledge axiom' = what any fool knows, any fool knows everyone knows.

Rules of inference of K4

1. Modus ponens.
2. If α is a tautology then infer $\boxed{s}\ \alpha$.

If this logic of knowledge, an agent's knowledge is represented as a theory called the 'agent's theory'.

Extending K4 to KI4

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

K4 is extended by the addition of 'circumscription modalities', which are written as $<\alpha>$ where α is a formula of K4. $<\alpha>$ stands for the α-theory of K4, hence $<\alpha>\beta$ is true iff β is in the α-theory of K4. These modalities capture the notion of 'provability' when the following logical axioms are added.

A6 : $<\alpha>\beta$ if $\alpha \Rightarrow \beta$ in K4

A7: $\neg<\alpha>\beta$ if $\alpha \not\Rightarrow \beta$ in K4

Circumscription is a form of default reasoning: the only proposition which are true are those that can be proved to be true. The circumscriptive nature of $<\alpha>$ derives from the restriction of its meaning to the α theory in K4. If α is a knowledge operator, then $<\alpha>$ may be translated as a circumscription of an agent's theory.

For example,

1. $<\boxed{s}q>$ ≡ the theory of agent s for which q is the only proper axiom.

2. $<\boxed{s}q \vee \boxed{s}r>\boxed{s}p$ ≡ the knowledge that s has about p is derived from the fact that he knows either
   q or r.

As an example the following wise men puzzle can resolved very well using this multi-agent logic.

A king wants to know who is the wisest of his three wise men. He paints a single white dot on each of their foreheads, and tells them that their own dot may be black or white but at least one dot is white. He then asks each man to determine the color of his own dot. After a while, the wisest announces that his dot is white. How did he know? His answer can be reasoned this way:

I see two whites on two foreheads. Suppose my dot were black, the second wisest would see a white dot and a black dot and would reason that if his dot were black, the least wise would see two black dots and would conclude that his dot is white. But that did not happen, and so the second wise must have a white. But that also did not happen and so the wisest must have a white. Such logic can be expressed and solved easily using Konolige's epistemic logic model.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Other knowledge base representations as production rules, slot and filler knowledge representations have been discussed later.

4.2.3 Rule based expert system

An expert system is capable of carrying out a difficult task which needs some degree of human expertise. Many of the past expert systems have been built as production rules based systems. One reason for that is expert knowledge can be easily coded as sets of rules. In simple terms, an expert system consists of:

- a database
- a rule-base
- a rule application module or a rule interpreter.

The database stores the data about the problem and the rule-base contains the production rules which look like *if* C *then* A where C is the antecedent condition and A is the consequent action and also all the facts. The action part A can denote different things as a command to retrieve the database, a command to accept data interactively and so on.

The rule interpreter module selects and applies rules which may yield changes and/or additions to the databases although other types of rule interpretations are also possible. In classical production systems the rule application module uses a data-driven approach to cycle through the rules looking for a condition that can be satisfied by the input data or database. The engine keeps on cycling till the goal is reached or a state is reached so that no rules apply (gives up). An example car starting-fault diagnosis system will possibly contain the following rules:

R1: *if* the engine does not turn on *and* the battery is not flat *then* ask the user to test the starter motor.

R2: *if* there is no spark *then* ask the user to check the points.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

R3: *If* the engine turns *and* does not start *then* ask the user to check the spark.

R4: *If* the engine does not turn *then* ask the user to check the battery.

.

.

Rn

Initially the problem (to be solved) is entered in the database which may be '*the engine does not turn*'. The engine cycles through the rules and finds the condition list of rule R4 is satisfied by the database and executes the action part of R4 requesting the operator to check the battery. Suppose the battery is not flat and a statement of this fact is also entered into the database which now looks as :

The engine does not turn.

The battery is not flat.

The engine continues again until either a rule with an 'exit' is executed or the database is unchanged after one complete cycle.

4.2.3.1 Search methods in Problem-solving

The above approach to problem-solving is said to be 'non-deterministic' due to arbitrary selection of the rule that matches. The selection will depend on in what sequence the data exists in the database and also the sequence of rules in the code. It is also called 'data-driven' since the database state is the sole identifier of the sequence of matching rules. There is another 'goal-driven' approach also where the engine starts with a goal and works backwards to see how the goal state follows from the original state.

4.2.3.1.1 Data-driven Approach

The data-driven approach is also called 'bottom-up', 'forward-chaining' or 'antecedent reasoning' approach. Here the rules are applicable only if their condition part is satisfied by the database. The following recursive procedure written in pseudo-Pascal outlines the structure of a simple interpreter that uses a data-driven approach.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
Procedure generate;
begin
  identify the set S of applicable rules;
    while S is non-empty
        do begin
          select a rule R from S;
          Apply R;
           if the problem is solved by the application of R then indicate SUCCESS
           else call 'generate' recursively
           remove R from S and undo the effect of applying R
        end
end;
```

The procedure 'generate' generates new facts from the database by using the rules. These new facts are then added to the database. The working of the procedure can be illustrated using an example. Suppose the rules in the rule-base are:

R1: *if* X is divisible by 12 *then* X is divisible by 6.

R2: *if* X is divisible by 20 *then* X is divisible by 10

R3: *if* X is divisible by 6 *then* X is divisible by 2

R4: *if* X is divisible by 10 *then* X is divisible by 5

Suppose, we have number N which is divisible by 12 and by 20, and the problem is to determine if N is divisible by 5. So initially the database contains :

N is divisible by 12

N is divisible by 20

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Procedure 'generate' will add two rules to S as S = {R1, R2} and apply R1. It will first propagate the substitution 'N for X' throughout R1 and then add more data into S as :

N is divisible by 12

N is divisible by 20

N is divisible by 6

Procedure 'generate' is called recursively with the applicable rules as: S' = {R2, R3} R1 is not included because it is currently being worked on.

In the new invocation, R2 is now selected and applied resulting the new database as:

N is divisible by 12

N is divisible by 20

N is divisible by 6

N is divisible by 10

The problem is not yet solved and so another invocation to 'generate' is made with the applicable database: S'' = {R3, R4}. R3 is selected and the database is changed to:

N is divisible by 12

N is divisible by 20

N is divisible by 6

N is divisible by 10

N is divisible by 2

Again, 'generate' will be called and the new set of applicable rules are S''' = {R4}

R4 is the only applicable rule which generates the solution. The rules that were used are R1, R2, R3, R4.

At this point the process continues and the system finishes the last recursive call by undoing the effect of R4 (N is divisible by 5) from the database, and then returns to the previous call that continues as follows:

- R3 is removed from S'' and the effect is undone i.e. the rule is removed from the database.
- Another pass-through the while loop still continues and as a result R4 is selected from S'' and applied.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Another solution is obtained. A trace of the rules used is:

R1, R2, R4 The process has still not finished; the effect of R4 is undone and since S'' is empty, the recursive call terminates that brings us back to the call procedure in which S' = {R2, R3}. The effect of R2 is now undone and R2 is removed from the database. The while loop continues; R3 is selected and applied giving the database state:

N is divisible by 12

N is divisible by 20

N is divisible by 6

N is divisible by 2

Application of R3 does not result in a solution and so the procedure is called recursively; this new recursive call commences by identifying a set of applicable rules:

S'''' = {R2} R2 is the only applicable rule and it is selected and applied; the database state is

N is divisible by 12

N is divisible by 20

N is divisible by 6

N is divisible by 2

N is divisible by 10

and the recursion continues and the procedure is called again; This recursive call commences by identifying a set of applicable rules:

S''''={R4}. This rule R4 is selected, applied and this results in a solution. A trace of the rules used in this solution is : R1, R3, R2, R4.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

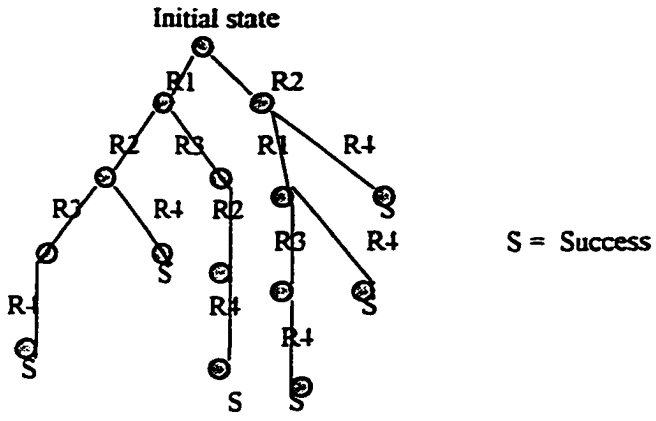


Figure 147:Search space tree

The process continues until all of the ways in which the solution can be achieved have been established. The 'search space' can be depicted by a tree as in fog 4-6 where each branch indicates the order in which applicable rules are selected and applied. This tree shows that:

- In the initial state only rules R1 and R2 are applicable.
- If R1 is the only rule that has been applied, then R2 and R3 are the only rules which are applicable.
- If R1, R2, R3 have been applied, then R4 is the only applicable rule and its application gives a solution.

(etc.)

In some applications (like scheduling etc.) it might be useful to identify the shortest path of all the solutions where all the solution paths need to be traversed. In some cases there may be more than one goal and one has to search the whole problem space.

Such an approach is simple, but may be inefficient and may appear aimless.

4.2.3.1.2 Goal-driven Approach

The goal-driven approach is also called the 'top-down', 'backward-chaining' or 'consequent reasoning' approach. In this approach, the system focuses its attention only considering rules that are relevant to the problem in hand. In this approach, the user begins by specifying a goal by stating an

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

expression E whose truth value is to be determined. For example, the expression might be 'N is divisible by 5'. The objective of the system would then be to establish the truth of this expression.

The following function, written in pseudo-Pascal, outlines the structure of a simple rule application module which uses the goal-driven approach.

```
function validate (X: expression) : boolean;
  var result: boolean;
   begin
      result := false;
      scan the rule-base to identify the set of applicable rules S which have X on the right-hand side;
      if S is empty
         then ask the user for some rules to add to S;
      while (result = false) and (S is non-empty)
         do begin
            select and remove a rule R from S;
             C := the condition part of R;
             if C is true in the database
               then result := true
                else if C is false in the database
                        then do nothing
                        else if validate ( C) is true
                           then result := true
          end;
        validate := result
   end;
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In order to start this process off, the user asks for the truth value of 'validate(E)', where E is an expression such as 'N is divisible by 5'. The first thing that the system does is identify all the rules which have E on the right-hand side assuming appropriate substitutions are made. These are the rules which might be able to establish the truth of E and are the 'applicable' rules in this approach. If no such rules exist, the system asks the user to provide some rules. If there is more than one rule satisfying the match, the system selects one.

When a rule R is selected, its left-hand side C is checked with the database. If C is true in the database (with appropriate substitutions), then this establishes the truth of E and the process can terminate with success. If C is false in the database, then R cannot be used to establish the truth of E and another rule is selected from S. If C is unknown (i.e. it is neither true nor false with respect to the database), then it is referred to as a 'sub-goal' and an attempt to establish its truth value is made by calling the function validate recursively with C as parameter. If validate (C) is true, then the rule R is applicable, the truth of E is established and the process can terminate with success. If validate (C) is false, then another rule is selected from S. Thus, the process works back from the goal trying to attain sub-goals which can themselves establish the goal.

As an example, again consider the production system with rules R1 to R4 as described above with a database containing the statements 'N is divisible by 12' and 'N is divisible by 20'. Suppose, the goal is to determine if N is divisible by 5. We begin by calling validate (N is divisible by 5). Execution of the function call begins by scanning the rule base, giving:

S = {R4}

i.e. R4 is the only rule with 'X is divisible by 5' on the right hand side (there was an effective substitution of N for X).

The process continues by selecting R4 and generating the sub-goal 'N is divisible by 5' by substituting N for X in the condition part of R4. Since 'N is divisible by 10' is neither true nor false in the database, the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

function validate is called recursively with 'N is divisible by 10' as parameter. This new recursive call begins by identifying the following set of rules:

S' = {R2} R2 is included in this set because it has 'X is divisible by 10' on its right-hand side. The process continues by selecting R2 and generating the new sub-goal (N is divisible by 20). Since 'N is divisible by 20' is true in the database, the recursive call terminates with validate (N is divisible by 10) = true. Thus, the initial call of validate terminates with value true.

Note that in general, goal-driven systems query the user as well as the database when determining the truth of a sub-goal. Thus, the user need not enter all of the data at hand initially, but may wait until the system requests data. For example, in the above, instead of entering (N is divisible by 12) and (N is divisible by 20) into the database at the beginning, the user could wait until the system has (N is divisible by 12) or (N is divisible by 20) as a sub-goal. The goal-driven approach may also be depicted graphically as in the tree in fig 148.

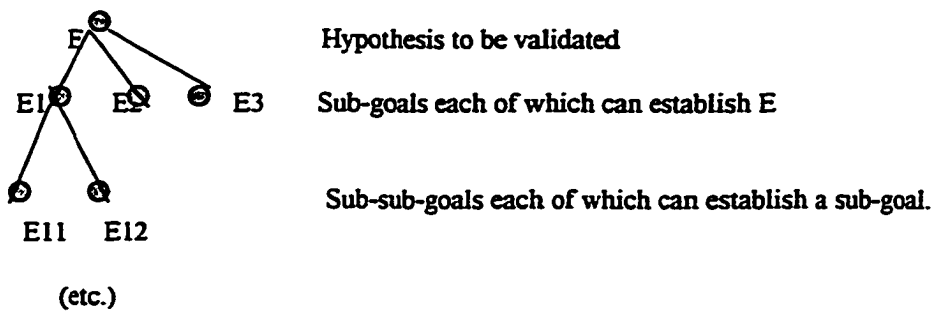


Figure 148:Search tree (goals) for goal-driven approach

The function 'validate' in the example terminates when a sub-goal is found to be true. However, if all of the ways in which E can be established are required, this can be achieved by changing the while loop control condition from ((result = false) and (S is non-empty)) to (S is non-empty).

Function 'validate' has been over-simplified by assuming the condition parts of the rules are single statements, not complex statement. In real applications, the conditions can be as:

if C1 *AND* C2 *OR* C3 *then* A

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The advantage with goal-driven approach is that it does not seek data and does not apply rules which are unrelated to the problem in hand.

4.2.3.1.3 MIXED METHODS

Data-driven and goal-driven approaches can be combined in many ways and one such example is given here.

Procedure mixed-method;

begin

repeat

let user enter data in the database;

call procedure 'generate' to generate new facts which are added to the database;

call 'select-hypothesis' to select a goal statement E;

call 'validate (E)';

until the problem is solved;

end;

The function 'select-hypothesis' uses the facts which are generated by the preceding step to select a goal statement which the system then tries to validate.

4.2.3.2 Search strategies used

Both the data-driven search and goal-driven search approaches are attempts to find a path linking the initial problem state to the goal state. In the data-driven approach the search moves from the initial problem state to the goal. In the goal-driven approach the search moves from the goal to the initial state. In both cases the system has to make decisions which have not been considered before. For example, in both cases the system has to decide which rule to select from the state of applicable rules at each stage. This decision is guided by a 'conflict resolution strategy', which is used by a scheduler. Other decisions

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

include decision how 'backtracking' should be done, and the order in which the search space should be explored: 'depth-first', 'breadth-first' or in an order determined by some rules of thumb called 'heuristics'.

4.2.3.2.1 Conflict resolution strategy

The way a rule is selected from a set of applicable rules is called 'conflict resolution strategy'. Examples of such strategies include those based on one or more principles:

(a) Select rules arbitrarily.

(b) Select a different rule in successive stages.

(c) Select the first applicable rule identified.

(d) Rules are given values which are used to determine which one to select.

4.2.3.2.2 Backtracking

If a rule R is selected from a set S of applicable rules and is used (i.e. in the data-driven approach it is executed and in the goal-driven approach it is used as a sub-goal), then one of two situations may be true after one or more stages:

(a) The system reaches a dead end, i.e. it gets to a stage where no new applicable rules can be identified.

(b) A path is established linking the initial problem state to the goal.

If (a) occurs, then in many applications it would be useful to be able to 'backtrack' to the set S, possibly undoing the effects of selecting R, and then to select another rule from S and use that rule in an attempt to identify a path.

If (b) occurs and only one solution is sought, then the process can end with success. However, if the user needs more than one solution it would be useful to backtrack as described above.

4.2.3.2.3 Depth-First/Breadth-First/Heuristic Search

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Depth-First Search

The data-driven procedure 'generate' and goal-driven function 'validate' that was outlined before may be categorised as 'depth-first' search methods. They traverse their respective problem-space tree in a depth-first manner, i.e. they start at the root, of the problem-space tree and work down the leftmost branch to the end-node (either success or failure) before they embark on any other branch. For example, given a problem space in fig 149, they would traverse the tree in the order indicated by the directed dotted line.

Problem space

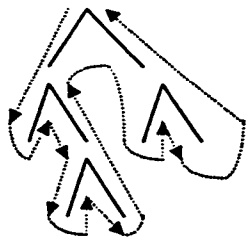


Figure 149:A depth-first search

In the case of the data-driven procedure 'generate', each branch of the tree depicts the application of a rule. In the goal-driven function 'validate', each branch represents a sub-goal. However, in both processes the search is depth-first.

There are two disadvantages of depth-first search:

(a) In depth-first search one can pursue an infinite branch indefinitely and never back up to the rest of the tree. This would occur if the rule base inadvertently contained a rule as :

if N is divisible by 5 then N is divisible by 5

(b) The depth-first search cannot guarantee to identify the shortest path to the solution before finding a longer path. In depth-first search one may find a solution, but another solution path may exist in part of the tree which has not been searched yet. This happened in the example of procedure 'generate': the first solution found had a path as: R1, R2, R3, R4, but the shortest path R2, R4 was the last path to be found in the solution.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The advantage of breadth-first search is its simplicity, and its memory economy.

Breadth-First Search

In 'Breadth-First' search, all of the nodes in the problem-space tree at level 1are developed first, then all the nodes at depth 2 are developed and so on. For example, given the tree in fig 150, the nodes would be developed in the order indicated by the node numbers.

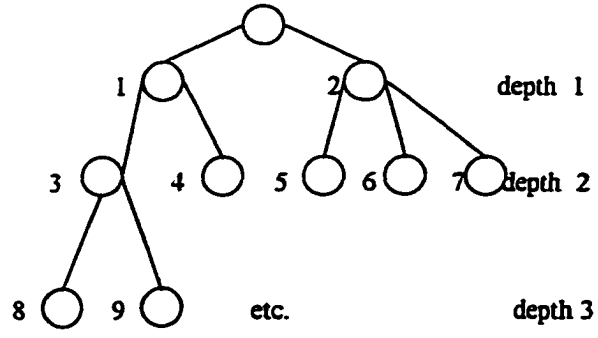


Figure 150:A breadth-first search

There are two characteristics of breadth-first search.

(a) All nodes will eventually be searched, given enough resources, even though one of the branches may be infinite.

(b) The first solution found will be with the shortest path.

Thus breadth-first search overcomes the problem of depth-first search. However, it can have extra overheads; at any stage all nodes to the left and all nodes must be memorised. In our example illustrating the use of the generate procedure, one would have to store the database state and a list of the applicable rules for each node that was developed if one needs all solution paths.

Heuristic search

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Depth-first and breadth-first search methods both search the problem-space tree without regard to the particular problem being solved. But humans do not solve this way. Generally they use metarules to decide which node to develop next. As example, in a bottom-up search the rules might be given weights indicating their proven usefulness for past problems. A metarule might then be used which states that if a choice exists, the rule with the highest score should be applied. The resulting search will neither be depth-first nor breadth-first. Such a search is called heuristic search. For a data-driven approach it will select the rule that has the highest weight. Heuristic search methods can also be used with goal-driven methods. In such cases the weight of a rule might be some value indicating the expected cost in achieving a sub-goal. The next node to be developed is always the one with the lowest weight.

4.2.3.3 Explicit representation of control knowledge

In the examples above the search strategy is coded as a procedure or function and the control knowledge is embedded in the code. There are some limitations on such approach.

- Once the strategy is coded in the programs, the flexibility becomes limited. But no particular mixture of data-driven, goal-driven, breadth-first or heuristic search strategy is ideal for every problem. Different problems need different approaches.
- Because the control knowledge is embedded in the code, the system cannot explain its problem-solving strategy; neither can the user easily modify the strategy for a particular problem.

There are good reasons to keep the control knowledge somewhere which is easy to modify. Some researchers have represented certain types of heuristic search strategy knowledge as metarules. An example of a metarule is :

if the infection is pelvic abscess *and* there are rules which mention enterobacteriacea in their premises

and there are rules which mention gram-pos-rods in their premise

then there is suggestive evidence (0.4) that the former rules should be applied before the latter.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Such metarules, which are invoked as part of the conflict resolution strategy, can capture and implement strategic knowledge about a domain. However, they leave out the domain-dependent and domain-independent strategic principles that underlie them. For example, the implicit strategic domain-dependent which underlies the above rule is that most frequent causes of a disorder should be considered first. The implicit domain-independent principle that underlies the rule above is that most 'useful' rule should be applied first. Clancey[84] discusses in his paper ways in which domain-specific knowledge can be related to domain-independent knowledge to produce more meaningful metarules.

Georgeff [85] offered a general production system architecture which contains the control knowledge represented using a control language. The architecture, which Georgeff calls a 'controlled production system', is based on the explicit specification of constraints on rule invocation. In Georgeff's scheme, control knowledge is represented by specifying a language whose set of wffs is the set of all allowable sequence of rules. A sequence of rule applications (i.e. a branch in the problem-space tree) is only allowed to be searched if the production rule is a wff of the control language.

For example, consider a production system with rules R1 to R7. Suppose that the control language were defined by the regular expression:

R1 (R2 R3 R4)* R5 R6 R7 where the * means any number of repetitions. The only allowed rule sequences are: R1, followed by the sequence R2 R3 R4 cycled any number of times, followed by the sequence R5 R6 R7. The restriction imposed by the control language can be invoked either when the set of applicable rules is being identified or as a part of conflict resolution strategy.

Ideally, search strategy control language should be encoded explicitly at two levels:

- domain-independent metarules
- domain-specific metarules

The domain-independent metarules may say as:

if no domain-specific metarules *exist* and memory space is limited

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

then

use depth-first search.

4.2.3.4 Uncertainty in Knowledge

In most of the domains in which expert systems are applied, the knowledge available may be unreliable, incomplete, imprecise, vague and/or inconsistent and one may have to reason with such incomplete, uncertain and vague knowledges. Nearly all existing expert tasks involve large search spaces, involve reasoning vagueness and uncertainty in knowledge and some involve reasoning with time-varying data. Various methodologies have been developed to reason with such knowledge. For example,

- Probability theory
- Certainty theory
- The Dempster-Schafer theory of evidence
- Possibility theory/fuzzy logic
- incidence calculus
- plausibility theory

The probability theory deals with knowledge about truly random events by assigning probabilities to the events. One important component of probability theory is Baye's rule which suggests a way of computing probability of a hypothesis being true for some given evidence to that hypothesis. Suppose the following:

(a) p(E/Hi) is the probability that evidence E will be observed given that hypothesis Hi is true.

(b) p(Hi) is the *a priori* probability that Hi is true irrespective of any evidence for or against it.

(c) k is the number of possible hypotheses which display evidence E. For example, k may be the number of diseases which display symptom E.

Baye's rule states that the probability that hypothesis Hi is true given evidence E is calculated as:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$p(Hi/E) = \frac{p(E/Hi) * p(Hi)}{\sum_{n=1}^{k} p(E/Hn) * p(Hn)} \quad .....(4\text{-}1)$$

An alternative expression of Baye's rule

Baye's rule can be expressed in an alternative form:

$$\phi(Hi/E) = LS * \phi(Hi) \quad .....(4\text{-}2)$$

where:

(a) ϕ stands for 'odds' and is related to probability as follows:

(i) $\phi(x) = \frac{p(x)}{1.0 - p(x)}$

(ii) $p(x) = \frac{\phi(x)}{1.0 + \phi(x)}$

(b) LS stands for 'likelihood ratio' and is calculated as follows:

$LS = \frac{p(E/Hi)}{p(E/\neg Hi)}$ where $p(E/\neg Hi)$ is the probability of evidence E when hypothesis Hi is false.

The LS value indicates the extent to which the presence of E supports the validity of the hypothesis Hi.

Complementary equations may also be written for the case in which E is absent. The equations are:

$$\phi(Hi/\neg E) = LN * \phi(Hi)$$

where $LN = \frac{p(\neg E/Hi)}{p(\neg E/\neg Hi)}$

The LN value indicates the extent to which the absence of the evidence E supports the non-validity of the hypothesis Hi.

Representing uncertainty of statements

a. Single value

The simplest method is to represent the uncertainty of a statement by a single value. In probability theory, this value stands on the probability of the statement being true. Initially, the value

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

stands for the a priori probability. Subsequently, the value is updated, using Bayesian rules, in the light of evidence relevant to the statement. There are some shortcomings to this approach:

(i) The single value tells us nothing about the precision.

(ii) This single value combines evidence for and against a statement without indicating how much there is

of each.

(iii) It is not always possible to obtain an accurate a priori probability.

(iv) Whenever a new hypothesis is added, many existing a priori probabilities will possibly require adjustment.

(v) Dependencies between statements must be known.

b. Two values: measure of belief and disbelief

An alternative approach is to maintain two separate values for each statement: MB(X/A) is a probability-like measure of belief in X given A and MD(X/A) is a similar value for disbelief in X given A. These two values are combined together into a single assessment of X given A, called the certainty measure C(X/A). This approach has the same shortcoming as (i) as above. However, (ii) is overcome since the measures for and against the statement are explicit.

c. Use of sub-intervals

A third approach is to present the probability of a statement by a sub-interval [l(X), u(X)] of the interval [0, 1]. Instead of specifying the probability of events exactly, one specifies the bounds meaning the probability of X lies between l(X) and u(X). The precision of the knowledge of the probability of X is explicit in this approach. This is the method used in the Dempster-Schafer theory of evidence.

4.2.3.4.1 Implication networks

An implication network is a directed graph in which nodes represent statements and arcs represent relationships between statements. An example of an implication network is given in fig 151.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Arcs between nodes are of two types: (i) those representing antecedent-consequent implications between statements, i.e. 'if-then' links, and (ii) those which are used to indicate that a statement is used as a component of a more complex statement. The former is called 'implication link' and the latter 'composition link'.

Implication networks can be used to indicate the relationships between evidence, production rules and hypotheses in an expert system. For example, the implication network in the fig 151 corresponds to the following production rules:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

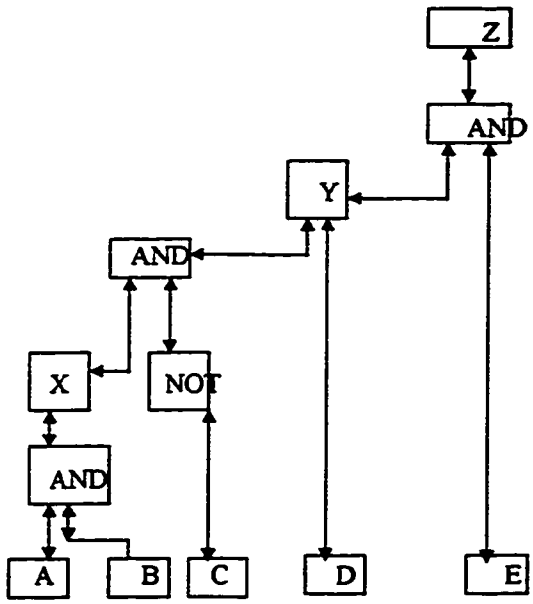


Figure 151:An implication network

if A AND B *then* X
if X AND NOT C *then* Y
if D *then* Y
if Y AND E *then* Z

Different values are associated with nodes and implication links depending on which theory (i.e. probability theory, certainty theory etc.) is being used in the particular expert system in question. For example, if probability theory is being used, the values associated with nodes represent probabilities and the values associated with implication links represent LN and LS values. If certainty theory is being used, then two values are associated with each node, one representing measures of belief and the other of disbelief and a single value is associated with an implication link representing a certainty factor.

4.2.3.4.1.1 Propagation of uncertainties in implication networks

The way in which uncertainty values are propagated through an implication network also depends on the theory that is being used.

(a) Probability theory

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

If the antecedent statement is definitely true, then the odds of the consequent are multiplied by the LS value of the link and the result is used to calculate the updated probability of the consequent. If the antecedent is definitely false, then the odds of the consequent are multiplied by the LN value of the link and the result is used to calculate the updated probability of the consequent. If the antecedent has some intermediate probability p, then the odds of the consequent are multiplied by some interpolated value determined by p together with the LS, LN values. A problem arises when a node is the consequent of more than one implication link; for example, node Y in fig 151. In such cases, correlation factors may need to be used.

(b) Certainty theory

Certainty theory was developed to deal with non-determinism. It was originally developed while developing MYCIN expert system. MYCIN needed to accommodate rules as "A suggests B' or "C and D tend to rule out E'. Certainty factors were associated with the action part of the rules in MYCIN as:

If (i) the stain of the organism is gram-positive, and

(ii) the morphology of the organism is coccus, and

(iii) the growth conformation of the organism is clumps,

then there is suggestive evidence (0.7) that the organism is staphylococcus.

The certainty value of 0.7 indicates that the evidence strongly suggests (0.7 out of 1.0) the conclusion, but not with certainty.

As mentioned earlier, evidence for a hypothesis is collected separately from that against it. The certainty of a hypothesis at any time (i.e. the certainty measure) is the algebraic sum of the measures of belief and disbelief in that hypothesis.

The calculation of certainty values of consequents of uncertain rules, when applied to uncertain antecedents, is carried out. In MYCIN, rules are only applied if the certainty value of the antecedents

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

exceeds an empirical threshold of 0.2. The certainty values of complex statements involving boolean operators is calculated as described below.

(c) Calculating the probability/certainty values of complex statements

Many expert systems, whether based on probability theory or certainty theory, use the following 'fuzzy' formulas for calculating values of combinations of statements:

cp (A or B) = max {cp(A), cp(B)}

cp (A and B) = min {cp(A), cp(B)}

whereas cp stands for certainty or probability depending on which theory is being used.

Use of possibility theory/fuzzy logic

A difficulty associated with all of the above methods are that they do not accommodate the notion of 'extent'. As an example, when asked the question 'how certain are you that you suffer from stress?', many people will answer 'a little' where they really mean that they certainly suffer from a little stress. Degree of belief is confused with extent along a subjective scale. Possibility theory/fuzzy logic accommodates the notion of extent along scales, where the degree of extent is related to 'vagueness'. Many new expert systems use this possibility theory/fuzzy logic.

Use of plausibility theory

Plausibility theory has an important role to play in expert systems due to the fact not all sources of knowledge are 100 % reliable. Some methods exists that allows plausible consequences to be deduced from inconsistent sets of assertions which have been obtained from sources of less reliability. Such techniques can be used to pre-process knowledge before it is used in an expert system.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.2.3.5 Features of Expert systems

Many expert tasks involve large search spaces. There are some methods to handle such problems.

Reasoning by elimination

In some applications it is necessary to identify all the solutions to a problem (like in medical diagnosis). 'Reasoning by elimination' offers a viable way of keeping the size of problem-space tree within limits. Here, the non-solutions (or solutions with low plausibility) are pruned from the tree as soon as possible.

Use of Abstraction

In certain types of planning problems, abstraction method can be used. 'Abstraction' involves identification of the important aspects of a problem and its subsequent partitioning into sub-problems.

Top-down refinement

In some applications of expert systems, the problem encountered cannot be partitioned into a 'standard' set of sub-problems. However, the principle of abstraction can still be used using many levels of abstraction. Higher levels are more abstract than lower levels. When the expert system has solved the problem in one level, it moves down to the next level, and then the next and so on. The order and content of levels is pre-defined, whereas the order is dependent on the task on hand.

Multiple lines of reasoning

When humans tackle complex problems, they use multiple lines of reasoning which may involve different representations of the problem. The problem solver takes advantage of the particular strengths of the different approaches. The problem is solved by integrating various lines of argument such that weak points in one line are covered by strong points in other lines.

Multiple sources of knowledge

Certain types of problem are best solved by a number of experts, each of whom uses his own knowledge to tackle a particular aspect of the problem. A blackboard architecture is a good example.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.2.3.6 Blackboard Architecture

Blackboard is a model of expert system that uses multiple knowledge sources and multiple lines of reasoning. The term 'knowledge source' is used here as it is common in literature. A knowledge source is a set of rules used by an 'individual' expert to tackle a particular aspect of a problem.

The notion of multiple knowledge-sources was first used in the HEARSAY-II system, which is a system capable of recognising spoken queries to a database system. The knowledge-sources in HEARSAY-II include, among others the following:

	Source	Function
(a)	SEG	Digitises the signal and produces labeled segmentations.
(b)	POM	Hypothesises the syllable class of a segment.
(c)	MOW	Hypothesises the word from the syllable classes.
(d)	WORD-SEQ	Hypothesises word sequences.
(e)	PREDICT	predicts words that follow phrases
(f)	(etc.)	

The knowledge sources communicate via a shared data structure called a 'backboard'. When a knowledge source is activated, it either creates a new hypothesis which it writes onto the blackboard or it modifies the certainty factor of an existing hypothesis. The hypotheses on the blackboard are arranged in two dimensions: level and time. The level dimension ranges from 'low-level' hypotheses concerning word segments to 'high-level' hypotheses concerning word the meaning of the whole sentence. The time dimension ranges from when the utterance started to when it ended.

The application of knowledge sources is controlled by an 'opportunistic scheduler' which combines a least commitment strategy with strategies for managing limited computational resources.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Different interpretations which occupy the 'same' location on the blackboard are called 'conflicting' hypotheses. For example, the blackboard might contain two word hypotheses for the same, or overlapping (in time), parts of the utterance. When conflicting hypotheses have the same certainty value, the scheduler [86] has to decide which knowledge source to activate next. HEARSAY-II uses a mixture of data-driven and goal-driven search. For example, data-driven search is used in attempting to synthesize phrases from temporally adjacent word hypotheses, and goal-driven search is used to reduce hypothesised words into alternative sequences of phonemes.

Comments
Many expert system shells are very popular today and they work in various domains. These expert systems come with shells and through an interface the user enters the domain-dependent rules. Human interface to expert systems is another area of importance which tries to simplify and enrich man-machine interaction.

4.2.4 Slot and Filler Knowledge Representation

4.2.4.1 Introduction

The languages of formal logic does not allow one to structure knowledge of the universe. One cannot get all the knowledges related to a single entity. The slot and filler representations reflect the structure of the parts of the universe also. These representations include semantic nets, frames, conceptual dependency structures and scripts. In 'frames' all assertions about one entity are held together. In 'conceptual dependency structures', all assertions about a particular action or event are held together and in 'scripts' all assertions about a particular sequence of events such as 'going to a restaurant' are held together.

The advantage of clustering assertions in this way is not only an improvement in access of knowledge from implementation point of view, it is often desirable to have related representations in one place. A more important advantage is that 'stereotype' structures can be represented. Such structures are called 'slot' and 'filler' structures, since they contain slots as 'name slot', 'parent slot' etc. which can be filled by values relating to a particular entity.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Slot and filler structure facilitate pattern recognition, inference of generic properties, handling of default values and the detection of errors and omissions in bodies of knowledge. An example semantic net is shown in fig 152.

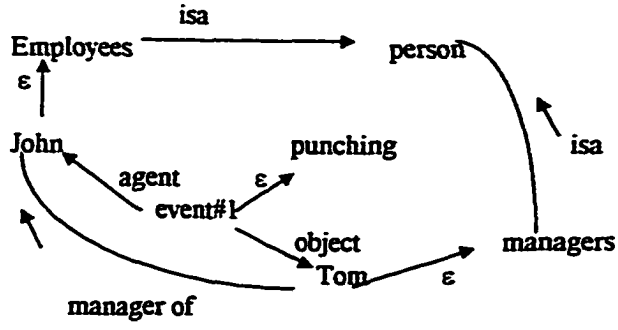
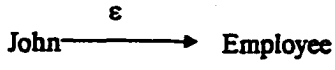


Figure 152:Semantic net for a statement 'John punched his manager'

4.2.4.2 Semantic Nets

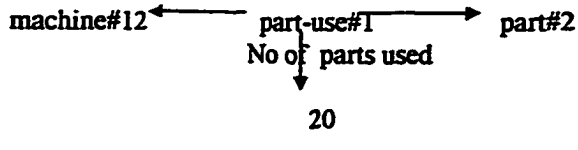
A semantic net is a directed graph in which nodes represent entities and arcs represent binary relationships among the entities. Arcs are labeled with names of the relationship types. A single entity is is shown as a node. Semantic nets can be used to contain various types of knowledge.

(a) Set-membership relationships may be represented by ε arcs, e.g.



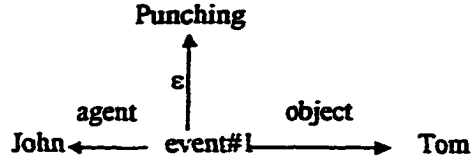
(b) Subset relationships are shown by isa arc.

(c) n-ary relationships (with n>2) can be shown by 'creating' an entity to stand for the n-ary relationship. For example, the fact that machine#12 uses 20 instances of part#2 in its construction may be shown as:



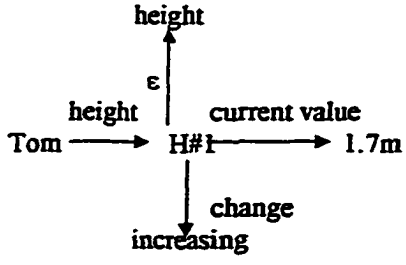
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(d) Events may be represented; for example:

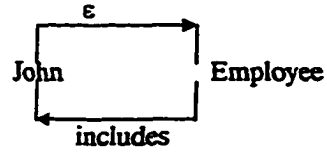


which represents the event of John punching Tom.

(e) Statements such as 'Tom's height is 1.7 meters and is increasing' may be shown as:



Note that the arcs are directed. If one wants to show facts in both directions, it must be do e explicitly. For example, to show 'John is an employee' and 'the set of employees include John' one will draw as follows:



These basic ideas of semantic nets have been adapted further.

4.2.4.3 Frame based Systems

Definition of frame

A frame is a data structure that represents an entity type. A frame consists of a collection of named 'slots' each of which can be filled by values or by pointers to other frames. Two examples of frames are shown in fig 153.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

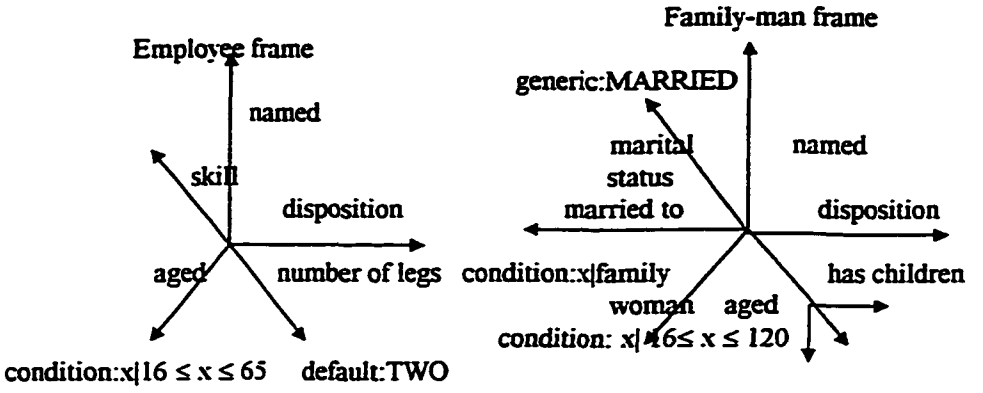


Figure 153:Examples of frames

Instantiation

When the slots of a frame are filled, the frame is said to be 'instantiated' and then represents a particular entity of the type represented by the unfilled frame. Examples of instantiated frames are given in fig 154.

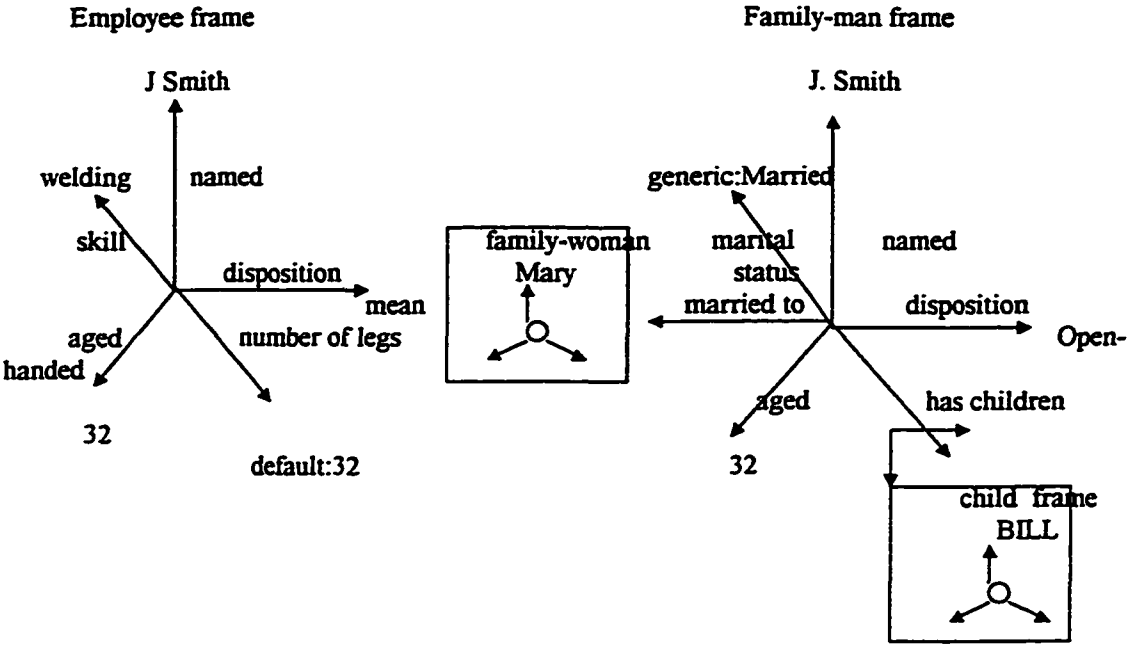


Figure 154:Examples of frame instantiation

Representation of an entity from different points of view

Frames may be used to represent the same entity from different points of view, as in fig 154 in which the person J. Smith is viewed as an employee and as a family man.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Generic properties

Pre-set values in a frame, such as the value MARRIED in the 'marital status' slot in the family man frame in fig 154 are called 'generic values' or 'generic properties'. They are values which occur in each instantiation of that frame. Thus, they correspond to properties which all entities of a particular entity type possess. For example, every family man has marital status MARRIED. (note that the family man frame in fig 153 represents one definition of 'family man' which may not match with another valid definition.)

Default values

Default values in a frame, such as the value TWO in the 'number of legs' slot in fig 153, are assumed values if no other value is supplied. For example, unless otherwise stated, one can assume that an employee has two legs.

Slot conditions

Conditions such as $x \mid 16 \leq x \leq 65$ in the 'aged' slot in fig 153 restrict the values with which a slot can be filled. Conditions may be more complex and may refer to other slots. For example, the allowed age of an employee will depend on whether the e employee is a male or a female.

Frame structures

One frame may refer to another frame and such references may go on and on or frames may be linked as entity sets. Frames may be inter-linked to other frames in the following ways:

(a) A slot in one frame may be filled by another frame (or by a pointer to another frame). For example, in fig 153, the 'married to' slot in the family man frame is filled by a family woman frame.

(b) Frames may be linked in taxonomical structures as illustrated in fig 155. The father/son links in such structures represent sub-set relationships. For example, the entity set 'employee' is a sub-set of the entity set 'person'. Sibling links which are shown as dotted lines represent sub-sets that have the same father. Similarity links shown by dashed lines are representing sub-sets which are similar in some specified sense.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

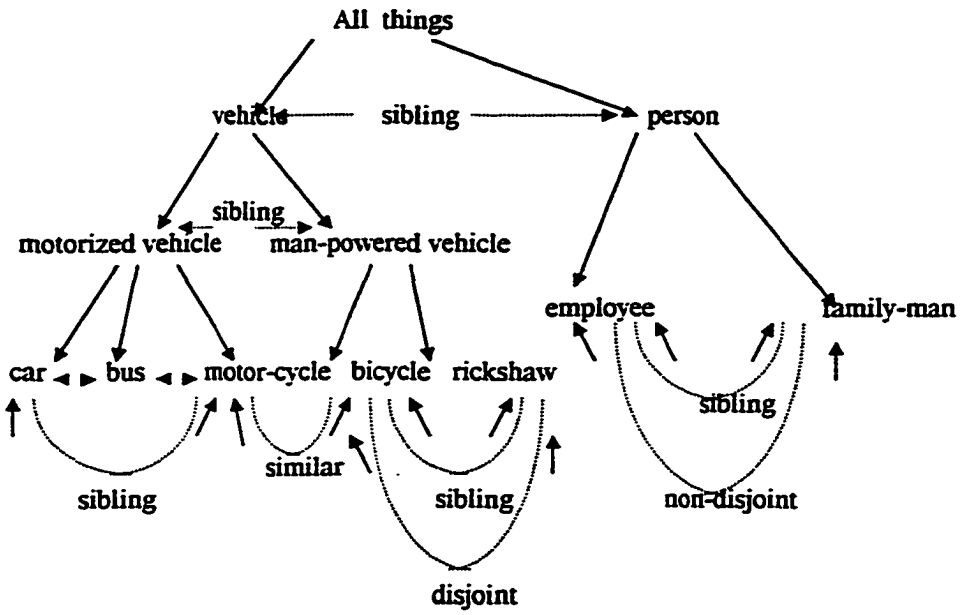


Figure 155:A frame-hierarchy structure

These links between frames can be used to speed up matching and inference of generic properties.

4.2.4.3.1 Frame instantiation and the representation of an entity

An entity may be viewed differently from different points of view. Suppose that we know the values of several properties of some entity E. If these values are appropriate to fill all of the slots of some frame F, then we can use the instantiation of F to represent E in which case E is being categorized as entity F. Suppose that from these properties of entity E, one can also fill out the slots of another frame H. In such case, one can also use the instantiation of H to represent E and in this case E has been categorized as entity type H. As an example, a person may be an employee as well as a family-man, seen from two different points of view. Also there are entity sets which are disjoint; for example family-man and family-woman. Inclusion of an entity in both the sets will be an error. The extent to which entity sets are allowed to intersect must be represented in the system in some form or other if such inconsistencies are to be detected.

Given that an entity may be instantiated in two non-disjoint entity-sets, inconsistencies can still arise which will need to be resolved. For example, the entity J. Smith has apparently inconsistent values

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

for the 'disposition' property when viewed as an employee and as a family man. There are some explanations:

(a) 'Disposition' at work and at home are really different concepts and should have distinct names.

(b) J Smith may behave differently at home and at work. However if this approach is adopted in all cases, one would miss out even the real inconsistencies such as a person having different values in 'aged' slot in alternative view frames.

(c) Such differences are errors.

4.2.4.3.2 Generic properties, Default values

Generic properties

A 'generic property' within an entity set is a property such that each member in the set will have the property. For example, the entity set 'person' may have the generic properties:

(a) Has blood temperature warm.

(b) Has some mother.

(c) Number of hearts is one.

Here properties a and c simply stated properties. But property b states it requires a value without specifying the value. The entity sets and generic properties are used very commonly by human beings. Whenever some property is incompletely stated, we try to match that with the available generic properties.

Default values

An example of a default value was in fig 153. A default value is one which is to be expected in the usual case unless stated otherwise. As example, most employees have two legs and unless there is contrary knowledge, one can assume this to be true. Although concepts of default values is important in knowledge processing, but sometimes use of too soon may complicate the problem rather than solving a problem As an example, suppose one says 'Jeane and Rich are dead in the living room.' If one assumes,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

they are humans one is mistaken. It is thought that many difficult problems can be solved if we deliberately prevent default values from blocking our thoughts.

4.2.4.3.3 Frame Structures

As was illustrated in fig 155, the frames may be related in various ways.

(a) Two frames 'person' and 'employee' may be related on a 'sub-set' basis or a 'isa' relationship

between them. Note that this sub-set relationship is between entity sets, not a set membership ε relationship. This means, all employees are persons or 'employee' entity set is a specialization of entity set 'person'.

(b) Two frames, 'motorised vehicle' and 'man-powered vehicle' can be siblings where they are children of the same parent frame. This means they are two sub-concepts within the same parent.

(c) Two frames may be related by the 'disjoint' relationship if they represent disjoint entity sets. Two disjoint frames can not be instantiated to produce alternate view-points w.r.t each other.

(d) Two frames may be related by the 'similar' relationship if they represent entity sets whose members have sufficient properties in common to be regarded similar.

(e) More relations can be added to frames. For example, a frame may exist to show the left-side view and another frame may exist to show the right-side view. These two entity-sets may be connected by 'rotation' relationship. More complex relationships can be added as in fig 156.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

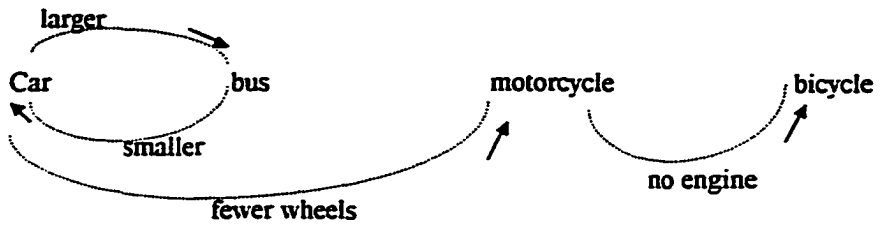


Figure 156:Example relationships between entity sets

The 'isa' relationship is a very important one because it defines the 'property inheritance path' which can go down and down transitively. For example, if a 'manager' is an employee and an 'employee' is a 'person' and a 'person' is a 'mammal' and the mammal entity set has the generic property 'blood temperature warm', then all instances of 'manager' frame will inherit this generic property. The 'isa' and 'similarity' relationships are important in concept matching.

4.2.4.3.4 Matching and pattern recognition

Frame systems are commonly used as components of pattern recognition systems, whether the patterns are visual patterns or abstract concepts. For example, a frame system may consist of a set of frames, one for each letter in the alphabet. The frame for letter 'E' may contain slots and values as 'number of straight lines =4', 'number of corners =2' and 'number of curves =0' etc.. Such a system may be used to recognize capital letters. In another frame system 'people' there may be many instances, one for male, female, father, mother, son, daughter etc.. Such a system may be used to 'recognise' the existence of a particular type of person when ever a role is mentioned in the text.

In some cases, when the given entity has matched a frame within a frame system, all its generic properties apply on the entity and this leads to inferencing. But matching being the first operation, is more important in finding the right frame. This means comparing a set of values associated with an entity against the values required to fill the slots of a frame. If all of the slots of a frame, F can be filled with appropriate values relating to some entity E, then F can be instantiated to represent E and E can be categorised as being of type F.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

However, F might represent a 'general' entity set such as 'person'. If we want to find a more specific match, one has to continue looking further down in the structure tree to see if a specialization of F can be instantiated. For example, one find that the values for entity E are appropriate to fill the slots of a frame G , called 'computer scientist'. Matching E with G is more informative than matching E with F. In some cases, one may not be able to find an exact match. Some slot values may not be known, some slot values may be outside the constraint range. In such cases, it will be necessary to find the best match. Here is the matching procedure:

Matching procedure

Suppose one knows a set of values of various properties of an entity E. The system can use these properties to select a 'candidate frame' as in the following ways:

(a) Start at the top level of the isa hierarchy and select the frame with the best 'match value' at the first level.

(b) Index the frames by slot name. Select any frame which has a slot name which is the same as any of the property names for which there is a known value for E.

(c) Select a frame arbitrarily.

(d) Use other contextual information to select a frame. For example, if it is known that entity E is a person, then select a person frame.

When a candidate frame has been selected using one of the methods above, the system partially instantiates the frame by filling as many of its slots as possible. In some applications, the system may ask for additional knowledge in an attempt to fill more slots. A 'match value' is computed that indicates the merit of the match. For example, if all slots are filled with appropriate values then the match value will be

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1. If only half the slots are filled the match value may be 0.5. The method of calculating the match value will differ from application to application. In some cases, the presence of an inappropriate slot value will set the match to 0. For example, an entity with 'blood temperature COLD' will have match value 0 with a person frame. In some cases, the inappropriate slot values may reduce the match value only partially.

Depending on the application, the match value will be regarded as signifying various degrees of success. If the match value is high enough, and the frame is specific enough for the application, then the system will stop looking further. But if the match value is relatively low, or the frame matched is not specific enough then the system will continue matching further frames. Upon finding a partial match, rather than looking at all other frame structures, the system can make use of the frame structure to identify the relevant frames in any of the following ways:

(a) The system can move up the 'isa' hierarchy until it finds a perfect match. It can then search the 'isa' sub-tree below the perfectly matched frame in attempt to find a more specific match. A depth-first, breadth-first or heuristic search may be used. For example, if one matches an entity E with an ape frame and obtains a match value of 0.3, one could move up the 'isa' hierarchy and find a perfect match with the mammal frame. One can then search the 'isa' sub-tree below the mammal frame in an attempt to match E with a more specific frame such as a dog frame.

(b) The system can look at frames related to the current one by the 'similar' or 'sibling' relation. For example, if an entity E matched the male human frame with a match value of 0.9 then the system could attempt to match E with frames such as boy human frame in pursuit of better match.

(c) Complex relationships between frames as illustrated in fig 156 can be used to identify candidates for matching. For example, if an entity matches the car frame except for its number of wheels which is TWO, then the relationship 'similar but fewer wheels' may be used to identify the motor cycle frame as a candidate frame.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.2.4.3.5 Inference

Frame systems use five types of inference: inference existence, inferred generic properties, inferred default values, recognition of abnormal situations and inference by analogy.

Inferred existence

When a match is made between an entity and a frame, the system is able to infer the existence of an entity of the entity type represented by that matching frame. The 'certainty' with which this inference can be made is related to the match value described earlier. It is not difficult to see how probability theory, certainty theory or possibility theory as used in production systems may be used in frame systems. A frame can be easily described by some production rules. For example, a person frame can be easily described by the following rules:

x is a person → x has blood temp WARM

x is a person → x has number of hearts ONE

x is a person → x has number of legs TWO with certainty 0.98

Inferred generic properties

When a match is made between an entity E and a frame F then the system can infer that E has all of the generic properties associated with frame F. The certainty of this inference depends on the match value between E and F and the certainty with which the generic properties are associated with F. The methods used in production systems for reasoning with uncertain knowledge could be adopted for use in frame systems.

Inferred default properties

When a match is made between an entity E and a frame F, and the value for some slot S in F is not known for E, then if there is a default value for slot S, the system can infer that E has this value for

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the property S. The certainty of such an inference depends on the match value and the certainty associated with the certainty value.

Handling of abnormal situations

Since frames describe general or expected properties of the entities belonging to some entity set, the absence of a value for a slot (or the presence of an inappropriate value) may signify an unusual situation. For example, if an entity matches a car frame exactly except that it does not have brakes then this signifies an unusual situation. Such knowledges should be flagged as important errors.

Inference by analogy

Following is an example of the use of analogy:

from (Bill is like a bulldozer)

and (a relatively high value in the weight slot of the 'bulldozer instantiated' vehicle frame)

infer (a relatively high value in the weight slot of 'BILL instantiated' person frame).

In analogical reasoning, one uses a statement as 'X is like Y' to infer slot values for X given values for Y. The following heuristics can be used in the process to select values of Y:

(a) Select values which are relatively high or relatively low compared with average values for that slot.

(b) Select slots which are important in the sense that they contribute critically to match values.

(c) Select slots which no close relative of Y possesses. That is, select slots which do not occur in frames which are related to Y by the 'sibling' or 'similar' relation.

(d) Select values which no close relative of Y possesses. That is, relatives may have the same slot but the value of this slot, which Y has, is not to be found in any of Y's relatives.

(e) Use all of Y's values.

To choose the property in X :

(a) Select slots that are not already filled in X.

(b) Select from those slots in (a) which are usually filled in X.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(c) Select all remaining slots from those selected in (a).

In this example of Bill being like a bulldozer, the use of heuristics was as follows:

(a) The bulldozer value of the weight slot is relatively high for the vehicle frame, there this slot is selected.

(b) Because no value for Bill's weight was known, a high value for the weight was selected to analogous with the high value in the bulldozer weight slot.

4.2.4.3.6 Frame languages and examples

KRL

KRL is a language that allows the user to build a frame systems. In KRL frames are called 'units'. There are three types of units:

(a) Basic units that represent disjoint entity sets, e.g. people, vehicles.

(b) Specialization units representing isa relationship, e.g. male person, employed person etc.. These units are not necessarily disjoint.

(c) Abstract units which represent abstract entity types, e.g. travel.

Instantiations of frames which represent individual entities are also called units in KRL.

[Person UNIT Basic

<SELF>

<Name (a String) >

<Mother (a Female_Person)>

< Age (an Integer)>
< Children's names (set Of (a String))>]

[Employee UNIT Specialization

<SELF (a Person)>

<Skills (set Of (a String))>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    <Pays Tax {(OR Yes No) Yes: DEFAULT}>]

[Female_Person UNIT Specialization

    <SELF (a Person)>

    <Gender = Female>


    <Husband> = (a Male_Person)]


[Female_Employee UNIT Specialization

    < SELF {(a Female_Person) (an Employee)}>]

[Person 1 UNIT individual

    < SELF  { (a Person with

                name = John

                Mother = Female_Person20

               Age = 35

               Children's names = (Items Jack Sue Peter)}>]
```

Figure 157:Some KRL units

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.2.4.3.7 Desired properties of Frame Systems

(a) The frame system should be able to efficiently identify the parents and children at any given time.

(b) The user should be able to easily enter a value for any slot in any frame at any time.

(c) The instantiation of one frame should be able to initiate the instantiation of other frames where appropriate, For example, the instantiation of a person frame with marital status slot value MARRIED should initiate the instantiation of a frame representing the spouse of that person.

(d) The system should respond as soon as an inconsistency arises.

(e) Analogical reasoning should be possible.

(f) The user should be able to specify how match values should be calculated.

(g) The user should be able to specify the extent to which a specialized match is required.

4.2.4.3.8 Advantages and disadvantages of Frame Systems

advantages

(a) Frames are intuitively appealing. The clustering of properties relating to an entity or entity set is 'natural' in the sense that people are quite familiar with the approach and use it in everyday discourse.

(b) Frame structures accommodate a taxonomy of knowledge.

(c) Default values are accommodated.

(d) Generic properties are accommodated.

(e) Procedural as well as declarative knowledge can be represented.

(f) Frames facilitate analogical reasoning.

Disadvantages

(a) There is yet no formal theory on frames. Hence the methods of inferencing and consistency checking are not based on a well-defined semantics.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(b) The mechanism for reasoning with uncertain knowledge (i.e. by use of match values) is based on *ad hoc* expressions with no formal foundation.

4.2.4.3.9 The relationship between frame systems and logic

There is much similarity between frames and logic as methods for representing and manipulating knowledge. For example,

(a) A frame with slot relationships R1, R2,.....,Rn representing an entity set E can be described using a formula of predicate logic as:

$$\forall x \in (x, E) \rightarrow \exists\, y1,..., \exists yn\ (R1(x,y1) \wedge R2(x,y2) \ldots\ldots \wedge Rn(x, yn))$$

(b) Generic properties in which values are specified may be represented as, for example:

$$\forall x \in (x, \text{Person}) \rightarrow \text{numberofhearts}\ (x, \text{ONE})$$

(c) The isa relationship can be represented as, for example:

isa (Employee, Person)

isa(Female_Person, Employee)

$$\forall x\ \forall y\ \forall z\ \ \in(x,y) \wedge \text{isa}\ (y, z) \rightarrow \in (x, z)$$

The last formula states that if an entity x is a member of an entity set y, and y is a sub-set of an entity set z, then x is a member of the entity set z. This enables an individual entity to inherit all generic properties of all its ancestors.

(d) Slot constraints can be expressed as, for example,

$$\forall x\ \forall y\ \ \text{aged}\ (x, y) \wedge \in (x, \text{EMPLOYEE}) \rightarrow \geq (y, 16) \wedge \leq (y, 65)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(e) Matching can written as:

$\forall x$ numberofhearts (x, ONE) $\wedge$ numberoflegs(x, TWO) $\wedge$ bloodtemp(x, WARM)

$\rightarrow \in$(x, Person)

(f) Classical logic cannot accommodate default values, calculation of match values or analogical reasoning.

However, some of the other logics can be used to describe them. For example, using a non-monotonic logic:

$\forall x \in$(x, Person) $\wedge$ Mnumberoflegs (c, TWO)

$\rightarrow$ numberoflegs (x, TWO)

$\forall x \forall y \forall z$ numberoflegs(x, y) $\wedge$ numberoflegs(x, z) $\rightarrow$ ident (y, z)

The first formula states that if x is a person and x having two legs is consistent with the knowledge base, then x can have two legs. The second formula states that person x can have only one value for numberoflegs. Hence if a person P has one leg then Mnumberoflegs(P, TWO) becomes false because in that case the knowledge becomes inconsistent (because of the second rule) and so the default value cannot be assumed. But if it is not known then Mnumberoflegs (x, TWO) is true and so the default value will be assumed.

Another difference is one which affects efficiency. In each theory of formal logic, the order of the formulas is irrelevant. Formulas relating to a particular entity set can be scattered throughout the theory. But in a frame system, much of the knowledge regarding an entity set will be mostly found in one place.

(g) Fuzzy logic can be used to accommodate inexact matching.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Finally, formal logic is a syntactic variant of frames. The entity sets and entities can be represented as unary predicates like PERSON(J.Smith) etc.. But still frames are easy to use and commonly used in text based systems.

4.2.4.4 Conceptual Dependency

The various methods of knowledge representation studied so far used various combinations of ontological primitives: entity, relation, entity set, attributes and function. Conceptual dependency differs in the basic set of primitives which include:

- various types of entity
- various types of action
- various conceptual cases
- various conceptual tenses
- various types of conceptual dependency

Schank[87] claimed that these are sufficient concepts of which everything in the world can be constructed which people talk or think about. In conceptual dependency theory the basic units of meaning are structured sub-sets of Scank's concepts which are called 'conceptualizations'. Conceptualizations can be linked with other conceptualizations to build infinitely complex structures. However, the basic building blocks are fixed and the way in which they may be interfaced is fixed by the rules of conceptual dependency theory.

4.2.4.4.1 Basic building blocks

1) *Primitive entity types*

There are four different primitive entity types denoted as follows:

PP - Picture producers or actors (i.e. the causers of acts)

PA - Picture aiders

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ACT - Actions

AA - Action aiders

Picture producers or actors are entities which perform acts. For example, if Bill jumped then Bill is a PP. Picture aiders are 'properties' of PPs. For example, if Bill is fat the 'fat' is a PA.

(2) Types of ACT

Various types of primitive ACT have been defined by Schank. They comprise:

ATRANS - abstract transfer, e.g. of ownership or responsibility.

ATTEND - focusing of a sense organ, e.g. to listen

CONC - to think about

EXPEL - expulsion from the body of an animate entity

GRASP - grasping of an entity, e.g. grip

INGEST - ingestion into the body of an animate entity, e.g. eat

MBUILD - to combine thoughts, e.g. decide

MOVE - movement of part of a body by its owner

MTRANS - to bring into the conscious mind

PROPEL - application of physical force, e.g. push

PTRANS - physical transfer of an entity from one location to another.

SPEAK - production of sound

Schank claimed that all verbs in everyday use can be analyzed in terms of these twelve primitive actions. For example, he claims that all the various verbs of thought such as think, dream, remember, conclude, decide, realise etc. can be constructed from CONC, MTRANS and MBUILD.

Acts are shown as double arrows linking actors to actions. For example,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Bill ⇔ INGEST. meaning Bill ate or drank something

A single verb in English may translate into more than one ACT in conceptual dependency. For example, the fact that 'Jack pushed Jill down the hill' would be conceptualised as 'Jack PROPELled Jill and this caused Jill to be PTRANSed down the hill'.

(3) Conceptual cases

Conceptual dependency theory includes four cases:

O - objective

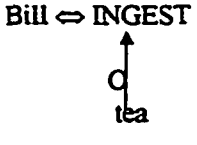
D - directive

R - recipient

I - instrumental

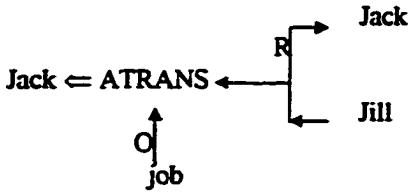
These cases are crucial because every action involves one or more of them. For example, in English one may say that 'Bill drank' or 'Bill drank a cup of tea' but one can only conceive of Bill drinking some entity which is the object of the action.

As example of the use of a conceptual case, consider the following representation of the fact that Bill drank tea:



As another example, consider the situation in which 'Jack took the job from Jill' which is shown as:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



(4) *Conceptual tenses*

The set of conceptual tenses used in conceptual dependency theory includes:

c - conditional ? - interrogative

delta - timeless / - negative

f - future

k - continuing

p - past

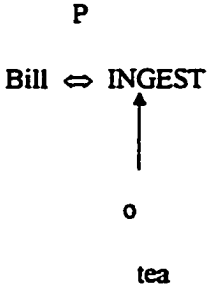
nil - present

t - transition

tf - finished transition

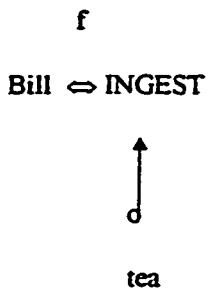
ts - start transition

For example, 'Bill drank tea' can be represented as follows:



whereas 'Bill will drink tea' is represented as:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Tenses may be omitted where the tense is irrelevant.

(5) *Conceptual dependency structures*

The basic building blocks described so far may be combined in various ways according to the rules of conceptual dependency theory. The resulting structures are called 'conceptualisations'. Conceptualisations can also be combined in various ways. The 'dependencies' between the basic building blocks in a conceptualisation are represented diagrammatically by use of arrows. The direction of the arrow indicates the direction of the dependency. The meaning of the word 'dependency' in statements such as 'A depends on B' ranges from 'A is only meaningful if B is present' to 'A is caused by B'.

The rules of conceptual dependency allow various types of dependency structure. The following examples describe some of the allowed basic structures:

(a) PP ⇔ ACT e.g. Bill $\overset{P}{\Leftrightarrow}$ MOVE

Actors and acts which are mutually dependent may be related as illustrated.

(b) PP ⇔ PP e.g. Bill ⇔ person

Entities and the entity sets to which they belong may be related as shown. The entity is placed on the left of the arrow and the entity set to which it belongs is placed on the right.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(c) PP e.g. Bill

A picture aider may be related to a picture producer as shown. The dependency is from PA to PP since, for example, the notion of 'fat' is meaningless without a PP.

(d) PP e.g. rabbit

Two PP's can be related in various ways when one is a 'property' of the other. The above depicts the conceptualisation 'the rabbit contained in the hutch'.

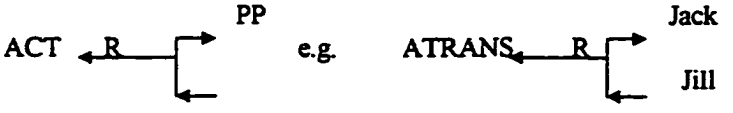
(e) ACT e.g. INGEST

An ACT may be related to the PP which is the object of that ACT.

(f)

An ACT which involves movement can be related to a source and destination. The example depicts the conceptualisation 'something was physically transferred from room 1 to room 3'.

(g)



An ACT may be related to the source and recipient of the ACT.

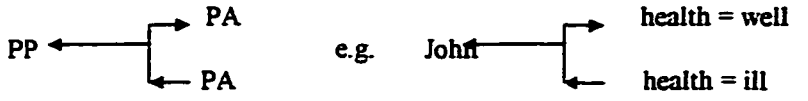
(h)

ACT ← I → ACT e.g. INGEST ← I → PTRANS

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

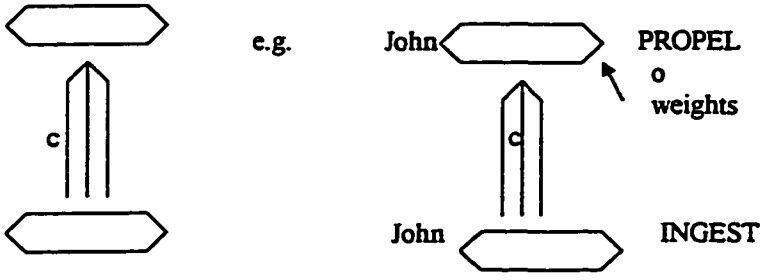
An ACT may be related to an 'instrument' with which the ACT was carried out. The example represents the conceptualisation that something was INGESTed by the PTRANSing (physically movement) of something else.

(i)



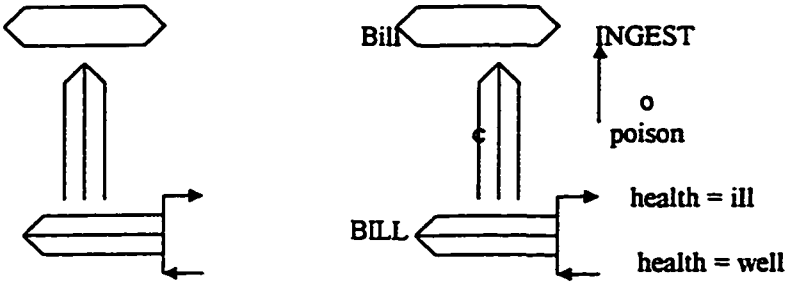
A PP can be related to a state in which it started and finished. The example represents the conceptualization that John went from being ill to being well, i.e. John got better.

(j)



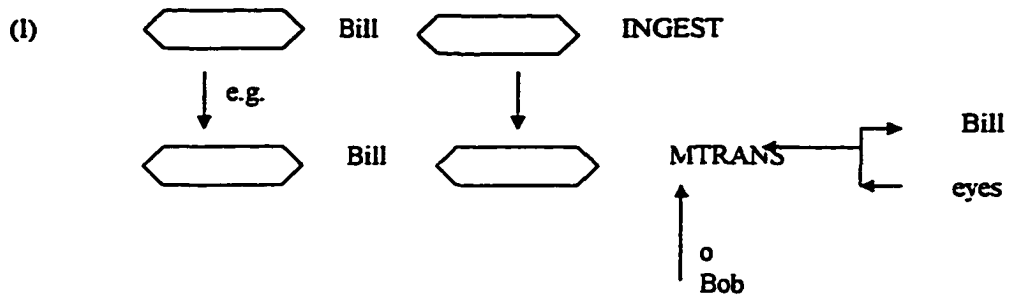
Two ACTS may be related where one caused the other. The example represents the fact that 'John INGESTed because he PROPELled the weights'; that is, he got thirsty or hungry because he lifted, pushed or pressed (etc.) the weights.

(k)



A state change may be related to an action which caused it, as illustrated. The example represents the conceptualisation that Bill 'changed' from being well to being ill because he INGESTed poison.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



Two conceptualisations can be related if one occurred at the same time as the other. The example represents the situation which may be described as 'while Bill was INGESTing, he saw Bob'.

There are many more conceptual dependency structures[87].

4.2.4.5 Scripts

A script is a structure which represents a commonly occurring sequence of events such as 'going into a restaurant, and ordering, eating, and paying for a meal'. An example of a restaurant script is given in

fig 158. This is a very simplified version of the original script.

Script name : restaurant

Roles : customer

waiter

cook

owner

Entry condition: customer is hungry

Props : food

table

money

Scene 1 : ENTER RESTAURANT

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

customer PTRANs into restaurant

customer MOVE into restaurant

goto scene 2

Scene 2 : ORDERING

customer MTRANS signal to waiter

waiter PTRANS waiter to customer

customer MTRANS 'Bring me food' to waiter

waiter PTRANS waiter to cook

waiter MTRANS 'Bring me food' to waiter

waiter PTRANS waiter to cook

waiter MTRANS (ATRANS food) to cook

cook DO (prepare food script)

go to Scene 3

Scene 3 : EATING

cook ATRANS food to waiter

waiter ATRANS food to customer

customer INGEST food

either return to Scene 2 to order more food or go to Scene 4

Scene 4 : customer ATRANS money to waiter

customer PTRANS customer out of restaurant

Results : customer is not hungry

customer has less money

owner has more money

restaurant has less food

Figure 158:A simple restaurant script

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A script contains slots which can be filled by the entities that are involved in an instance of the sequence of events represented by the script. Typically a script has:

(a) A set of *entry conditions* which must be satisfied before the script can be instantiated.

(b) A set of *roles* which are slots for people who would typically be involved in instances of the script.

(c) A set of *props* which are slots in the script for objects typically involved in instances of the script.

(d) A set of *scenes* each of which represents one of the events making up the sequence of events represented by the script. A scene may be represented by a conceptual dependency graph also.

(e) A set of *results* which will obtain after the sequence of events have completed.

Reasoning with scripts

A script may be activated by matching its name or the pre-conditions, roles, props etc. or by matching some combination of these properties. Once a script has been activated it can be used to infer, through default reasoning, knowledge that has not been given explicitly. For example, if one knows that Bill PTRANSed himself into a restaurant and that Bill ATRANSed money to a waiter, then we can infer that Bill was hungry, went into restaurant, ordered and INGESTed some food, and left the restaurant with less money than he has before.

4.3 Information Network

Information Networks are an abstraction upon intelligent network.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.3.1 Basic Concept

The trends of telecommunications applications as distributed office, video conferencing, home banking etc. are gradually requiring an integration of telecom services and computer applications. As computer applications become popular for the customers, these applications have to be more communicating & overlapping, flexible & intelligent. Distributed applications with embedded artificial intelligence will suit such requirement for future[8].

An information network will have various applications built into it once they are installed on the network and those applications will form part of the network's services. This information networking architecture embodies a distributed application platform for building and executing these network-based applications and the platform will hide the complexity of distributed nature (e.g. location, architecture, memory configuration of the host) and underlying network resources. The levels of abstractions are shown in fig 159.

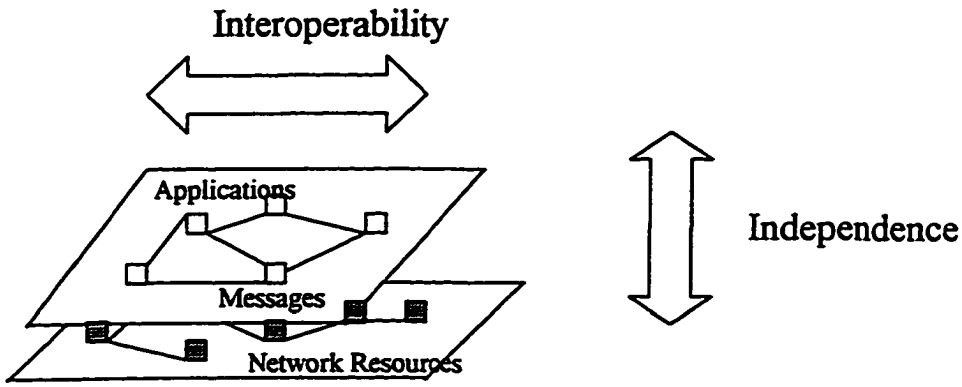


Figure 159:Platform model for Information Network

4.3.2 Objectives Of INN:

Internally, such an information network must meet a set of common standard requirements for being able to let distributed applications run:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1) It must provide a framework for distributed computing with its own distributed protocol (e.g. ADA's task-based distributed protocol, RPC, DCE, OLE 2.0, CORBA etc.) so that applications can run in this environment seamlessly. The applications will be distribution-transparent (as long as they follow the distributed protocol) thinking all the applications are local on the network node.

2) It will provide control of multi-session & multi-point communications using its underlying broadband network resources (e.g. databases, switching etc.).

3) It will provide services of inter-operability across diverse network domains with varying resources & features. In that case, the services will be universal irrespective of location & time.

4) It will provide rapid application creation and flexible deployment of complex services, including management of service creation. From some central locations administrators should be able to roll the applications into all the nodes of the network. The user-network interaction should be sophisticated and evolving, letting the user monitor and control any resources.

5) It will support widely diverse applications covering multi-media, motion pictures etc.. These network-based applications supported on INN will have to be modular, incremental, message-based, interface-driven, reusable and adaptive to be able to find out what resources (mainly network resources) they have available. Also each basic application unit has to be of limited complexity for easy maintenance. These applications will tolerate networking environment when they are interactive and cooperating among the distributed partners. They will need to encapsulate their own processing details to hide those from other units of processing. Other larger grain-size applications will be close to the distributed services of the distributed operating platform. Although AIN architecture does not clearly emphasize a general platform for service software, but it is apparent that object-oriented distributed architecture will form the IN architecture tomorrow and broaden its applicability.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Objects form natural boundaries for distribution of software on the network (as communication unit, recoverable unit. DB-access unit, processing unit) for their separation of interface from implementation and these objects of the network.

4.3.3 Evolution of IN to INN:

Today's IN architecture can be extended further to create a platform for object-oriented applications running in a distributed environment. Today's IN call-model can be modified to imbibe object-oriented principles and changed into some connection model built using participating agents from various call-leg stations. In any case, such objects will be the fundamental elements of processing exchanging messages among themselves (as permitted within the protocol of distributed operating environment). The distributed environment will be responsible for accepting and distributing these messages to the respective partner at the respective node.

For an efficient handling of message-distribution, the distributed processing environment will like to keep a list of all valid messages supported by all the objects existing in the network. For this reason, all the application objects need to register the messages they can accept (& the interface for the messages) I each network node. (Even the network administrator will be able to find out on the monitor which messages are up and which ones are down at the network nodes)[9]. These objects will participate in as many services as possible thus enhancing reusability.

Thus IN architecture will evolve into an object-oriented [12] software architecture filled with servicing objects while retaining the existing elements of AIN and various levels of abstractions in service formulation of IN and will be based more on the principles of distributed computing than purely on the network resources (this is just a broadening of the classes in the o-o paradigm).

4.3.4 Fabric of INN:

The information network will be a collection of objects each providing its service. The objects will have their own classes and one will be able to create as many instances of any class (unless there is

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

any restriction on concurrency). These classes may be arranged on a hierarchical inheritance tree in which case message inheritance occurs. Some objects may be computational in nature & return some values (e.g. shortest path between two stations or minimize digital cross-connection etc.) or they may be non-computational so as to set up a connection for a 3-party call.

Each service is an instance of a service type and there can be any number of them. Each service may be categorized by the quality of service, cost of service (physical resources they hold) etc..

4.3.5 Major elements of INN:

The information network will eventually become 1) a distributed operating system for object management with its own services and 2) a large object library with a huge blackboard of the message-interface of all the objects. The important requirement will be increased abstraction in classifying these objects. As a whole, it will offer a distributed processing environment (DPE) supporting many different services on various kinds of resources & these resources will be synonymous with the broad classes of the existing objects. Thus, packaging these objects into clusters will be important and as more packages will be added, the DPE will become richer with functionalities. Looking at it from inside, the resources of DPE will be the network resources which the application objects will use. The packaging may follow various principles as: separation of network transport functions from value-added service functions, further classification of service functions depending on their types, separating data-processing functions from business-rule processing, separation of company-wide data management operations from business rules, separation of user-interface functions from data-processing functions etc.[8]. Fig 160 shows an abstraction of such objects.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

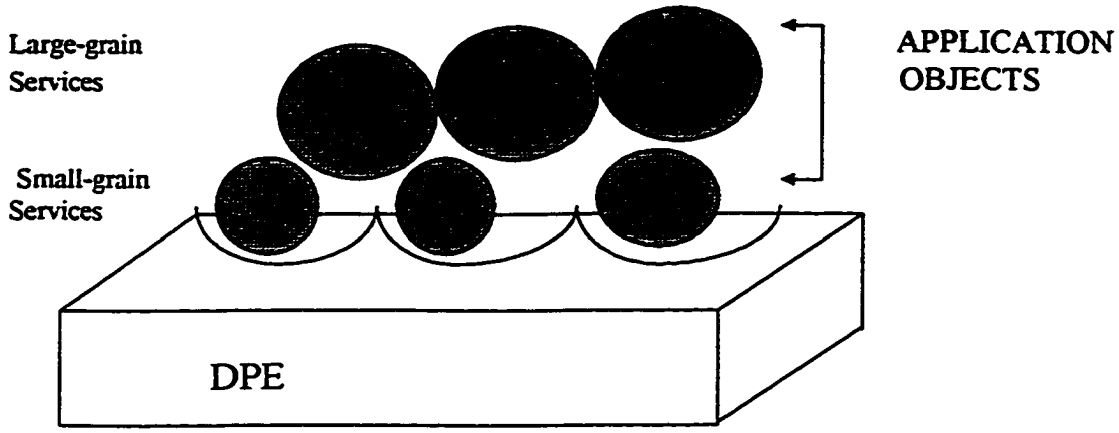


Figure 160:Abstraction of INN objects

4.3.5.1 Building Blocks

Such large packages of objects are called building blocks (containing a few computational objects) that will form units of cooperation on the network.

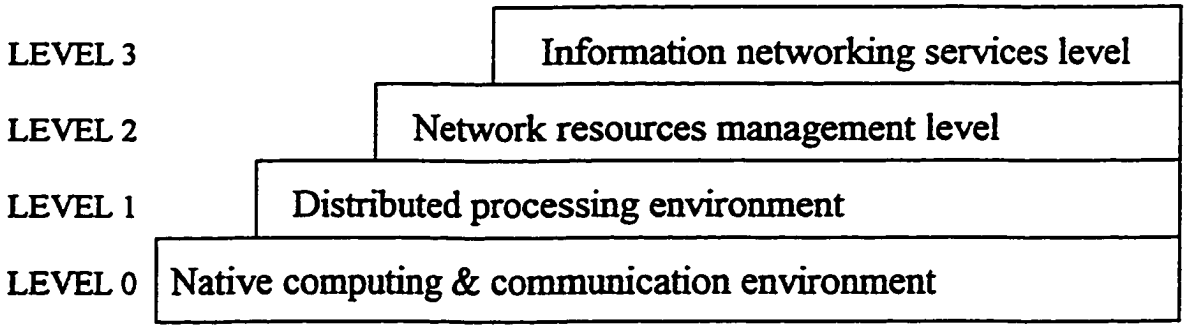


Figure 161:Architecture layers for information network

The messages which such packages want to advertise to the outer world so that they may be used from outside the package are called contracts. Two objects in two building blocks can communicate using an established contact only. This building blocks form unit of operation, unit of distribution, unit of security and unit of interoperability in the DPE and they declare their interface for the outside world. The objects may be classified into blocks by the type of their functionalities. For example, objects responsible for transmission & switching functions (e.g. connect/disconnect to a port or a socket, set up a VC etc.) can

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

belong to a delivery segment block and in this way more layers can be built upon the DPE layer. One such layering scheme is shown in fig 161.

4.3.5.2 Distributed Processing Environment (DPE):

The DPE services can be broken down into 2 types. 1)DPE kernels and 2) DPE contract services. The DPE kernel is responsible for transaction initiation, transaction termination, contract failure management, resource commitment, processing of remote procedure calls (RPC), processing of real-time programs etc.. Contract services include mainly contract registration, contract selection, contract validation. Contract access control etc.. Whenever a building block is installed in the system, all its defined contracts will have to be registered. At this time the registered valid contract-interfaces are added into a directory of supported contracts. Similarly, when a building block wants to find out if a contract exists in the system or not (or which contracts exist for a defined class of service), it can follow the method of contract selection and find out more about the contracts. It can search on contract name, class of service name or type of contract etc.. Authorization service provides access control checks for the contracts for use, installation etc.. Validation service only checks to see if the referred contract or block exists or not (this is done at the time when the requesters issue a bind for the server). As an example of such an architecture, one may choose the service interaction decision for a call/session over a heterogeneous network. In this case, the call initiator may subscribe to certain services which may not be supported by the carrier network at present time and similarly at the call terminating point. Also certain services will have to be discarded because of a conflict at the terminating point etc.. At present in IN, the services are controlled by a feature manager who decides the chosen features. All the possible interactions are pre-coded within the feature manager and the feature manager chooses by going through a match procedure.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

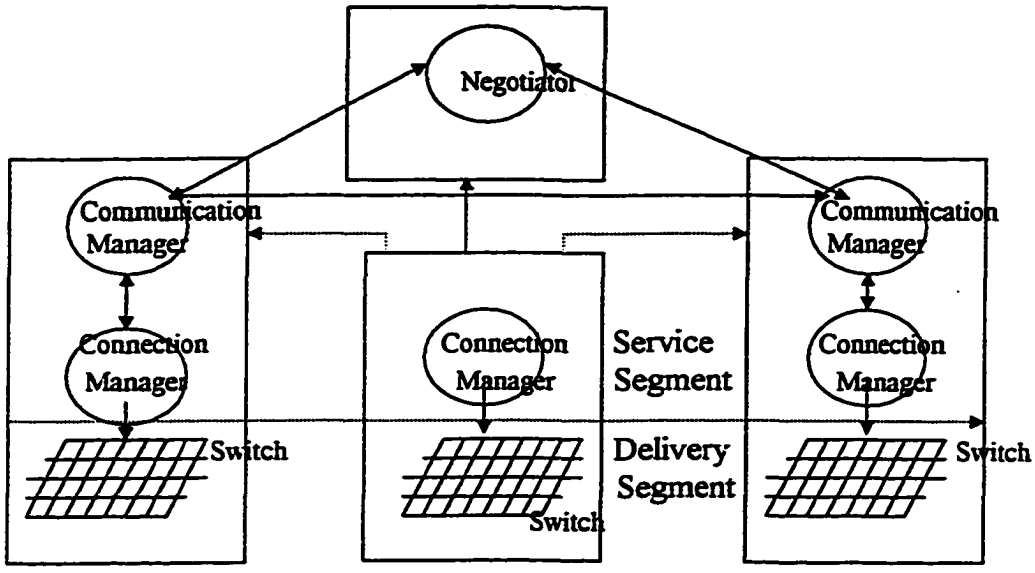


Figure 162:Service interaction architecture

But such a list can grow very large as more services get added and also a real time mechanism for decision making will be preferred. An alternative to this is 'negotiation of services'. Fig 162 shows a proposed architecture. At each point, there needs to be a connection manager which will work closely with the network resources and find out a list of services currently supported on the network and will be responsible for issuing all the commands for the network resources. There will be a transient object called communication manager (sitting above connection manager) which will decide the services to be added on the call and then it will go away. In the beginning, the connection manager building block will come up at both ends and intermediate points, initiate the basic call and then ones at each end will initiate a communication manager. The communication manager will receive a list of services supported on the local network currently and then register a contract for each service. The communication manager at each end will be the negotiating agents and they will have a path of communication. The communication manager at the calling end will initiate a negotiating object called negotiator which will find out all the services supported by the participating agents and decide the list of services to be used by emulating a common negotiating scheme.

Once the services have been determined, the communication managers & negotiators will go away. Such a method will be flexible and data-driven. The information network can in long-run support

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Once the services have been determined, the communication managers & negotiators will go away. Such a method will be flexible and data-driven. The information network can in long-run support database queries, multi-media applications & multi-media communications and will present an unified perceptions to operations and services.

4.4 Distributed Artificial Intelligence

4.4.1 Introduction

Distributed artificial intelligence (DAI) offers to solve certain problems in telecommunications through its ability of negotiating among a few distributed processes when a problem occurs. Such distributed models of solving the problem works well to solve the problem of feature interaction, restoration of a lost logical link and others that are commonly encountered in telecommunications. DAI shows how processes on independent computer systems can cooperate to solve problems that need their combined knowledge and processing effort. This becomes important when no single system has the complete control on the problem. DAI specifies how the computers reason about when to interact and what information to exchange among the distributed processes.

The use of DAI techniques in today's intelligent telecommunication systems appears appropriate due to its trend towards distribution of functionality and incorporation of intelligence and generality.

4.4.2 Principles of Action

Each partner program that makes up a DAI system has its own objective and more or less independent activities - is called agent. These agents has its search space defined for solving a problem on basis of rules or some other formalisms. The agents can be launched in the system in many ways. For example, each feature may be represented by an agent. So when a user makes a call, the call goes through as many agents as features the subscriber has. But a better distributed representation will be where each participating entity (subscriber or service provider etc.) will be represented as an agent. If each service

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

provider will be an agent, then many will be required to set up a session. No single agent will have complete knowledge of the dynamic network but during runtime these agents have to engage in a limited-domain search to find out an acceptable way of setting up the session. These agents need to be aware of the others before they interact, they must interact in a cooperative way to find a solution (i.e. they must negotiate). The agents heterogeneous, objective-driven and encapsulated. The agents make demands from each other and some may not be met and hence alternatives have to be searched and this way a solution is reached. Normally the agents communicate using a blackboard framework. At least by posting a message on such a framework when they start up is the way to inform all the other agents that they exist. Similarly before they come down, they post a message (e.g. ending) to indicate their nonexistence.

There are a few different models of agent-search space protocols existing today that have been tried in telecommunications problems. Two important ones are: 1)Multi-stage negotiation and 2) Negotiating agents, although other designs also exist.

4.4.2.1 1) Multistage negotiation:

Conry et. al.[5] used this protocol in one telecommunication application - restoring alternate physical routes when some physical links go down. Allocation of alternate physical paths keeps the logical links up and the objective should be to restore the entire bandwidth. A telecommunication network is divided into segments and each segment may be assigned to an agent existing somewhere on the segment. Now it may be assumed that the agents came to know about the neighbors when the physical circuits were initiated. When the network circuit path failed, the agents must detect it. After it observes a broken circuit, it constructs alternative plans to restore the connection and associates ratings with each of them. Each plan forms a hierarchy of plans that can achieve the goal. Thus it forms an AND-OR tree of subgoals (AND-ed) and strategies (alternate). This requires dynamic information of the network and hence needs to be built at runtime. Once all the local plans are produced, a global plan is found that can restore most of the routes through cooperative interaction among partner agents.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

There are three steps in this. In the first phase, the agents exchange information of their local plans. So each agent decides the demands from their neighbors in the local plan and checks the possibility of conflicts. The conflicting plans are discarded and surviving plans are joined together looking for further conflicts. If a global plan does not come out of this simple search, then the second step begins. Here the agents represent the non-local conflicts abstractly and share knowledges among themselves trying to reach a decision. If the problem is still found over-constrained, the third step begins. In the third step, the constraints are relaxed (looking for a partial solution) in a systematic order. The local goal tree is updated with new goals and a solution is sought. This sequence of steps (1, 2, 3) repeats iteratively until the whole problem is resolved.

4.4.2.2 Negotiating Agents:

Certain categories of conflicts for setting up a communication session may be pre-anticipated and this model provides a way of working on such conflicting situations. The conflicts may be on some specific details of the session. Options & features demanded by multiple users at the stations etc.. To resolve these conflicts an agent may be launched to represent each user and the agent knows its goals, priorities and policies. These agents negotiate cooperatively to determine an acceptable specification for setting up the communication session. Information regarding what is required for the session is represented formally and maintained locally and there are restriction on information that can be given to a partner agent.

Each agent has a goal hierarchy which represent the various ways sessions may be set up. An entity then marks the goals in the hierarchy to indicate which are acceptable and which are not. The hierarchies contain two types of relationships 1) specialization and 2)decomposition (sub-goals). A goal is achieved if any of its specialization is achieved or all the sub-goals are achieved. Agents create plans that achieve a particular goal recursively with one of its specialization or all of the components. The agents prepare such plans and transmit to the partner agent. The partner agent tries to match that on its goal tree

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and replies if it is acceptable or not. Such exchange continues unless a plan is acceptable to all partners. This it finally reduces to a tree search.

There are many more schemes for negotiation among agents with DAI more details of which may be found in references [5, 6].

4.5 A Network-Based Information System (NBIS)

4.5.1 Introduction:

A good application of information network can be a network for distant knowledge acquisition. Nowadays network-based learning systems carry course presentation, classroom materials, supporting bibliographic library information retrieval or bi-directional movement of multi-media are emerging. IN provides a good starting ground because of its distributed architecture, programmability, high bandwidth capacity and extensibility of its signaling protocol.

Today most of the bibliographic & full-text data-bases of the libraries of the national & international universities are already connected on the Internet. In addition, the existing IN provides us fast connectivity to the existing SCPs and switchability. A knowledge-based information system (KBIS) is a generic network system (that uses both IN and public data networks) basically accepts queries from a remotely located student or a learner and fabricates its response from the vast units of its knowledge-bases.

This network uses its communication capability (which is adapted from IN) to collect & compile queries, then routes the query to a "knowledge-ring" comprised of the knowledge-bases arranged in some fashion and then bursts the responses back from the knowledgeable instructor. In addition, it possesses a language interface which accepts the user's query, comprehends & rewrites the query in the context of the existing knowledge-bases (reformats or adds other elements) and then restates the query back to the user

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

asking for his approval. The user may change it, approve it or decide to restart from scratch. The approved query will be sent to the respective knowledge-base using the fast IN switching path (and using the signaling protocol) for response. Such a method will suit an interactive researcher as well.

This NBIS is an extension of IN and so it can be a gigabit speed network. In addition, it is totally compatible with any high-speed campus network as ATM network (with SONET equipment) etc.. It is worthwhile to mention that the concept behind NBIS can spread far in future although only a part of it may be implementable today.

4.5.2 Concept behind NBIS:

Ideally speaking, NBIS represents an educational knowledge server answering any question from a questioner in a continued interactive session. It can search documents on a bunch of network hosts and retrieve the results. To add more functionality, it can even use reasoning among successive question and find out where the logical progression of questioning did not take place, where student changed the track altogether. But such analysis goes into the area of context-based semantics analysis & understanding, linguistics analysis (very little of which has been achieved today) requiring almost infinite data-base of world-knowledge for making associations and a very small version of it can be realized today.

4.5.3 Role of knowledge-base:

It is known that knowledge is dynamic, quantifiable and classifiable. Hence if the context-based semantics for a question is understood then any text related to the knowledge sought can be retrieved from the texts in the storage systems. The knowledge-bases are storage areas of such knowledges.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.5.4 Description of NBIS:

Although this network resembles IN, but it has many new elements. The major new elements of this network can be classified into 3 classes: 1) basic existing elements, 2)other supporting elements, 3)key new elements[21, 22].

4.5.4.1 Basic existing elements in the network:

These elements are already existing in some form in IN.

4.5.4.1.1 Campus Network (CN):

NBIS is an intelligent network that interconnects many campus networks. The campus networks can be based upon any protocol providing the user access points for NBIS. The user access points are workstations which accept the users' queries and send them over to query processor lying outside the campus network on the intelligent network of NBIS. Similarly the responses from NBIS will be received in the CNs. The CN will include network nodes for NBIS or will contain gateways to connect to the IN of NBIS[21].

4.5.4.1.2 High speed Data Networks:

There will be some front-end processing (compiling) elements on the national (or international) public data networks that will collect the query packets, participate in interactive understanding of the queries and finally will compile the queries. Although circuit-switched telephone networks may be good candidates for this, but their limited channel capacity may choke the speed of the query packets. The new evolving networks (B-ISDN with OC-12 channel etc.) will be better candidates for this. Finally the compiled query packets will be sent to the backbone network which can be an IN. Similarly, on the way down to the campus networks from the knowledge-ring, the response packets will need to be buffered and routed through these processing elements. It is important to note that high speed of transmission is essential for the up-path of the query packets whereas the response packets can be carried back at slower speed (as to match the user's viewing speed) and hence proper bandwidth allocation should be done for these paths.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.5.4.1.3 Knowledge Server (KS):

Each knowledge server represents the modules of knowledge which is used to answer users' queries. Each of it may be specialized, current and concurrently accessible at very high speed. Such data-bases may be connected or may be left independently isolated. There are many structured schemes or networked schemes to represent such knowledges (e.g. semantic nets, conceptual graphs, conceptual dependency, scripts, frames or objects etc.) [10].

4.5.4.2 Other supporting elements:

These are some existing elements of the network.

4.5.4.2.1 Knowledge Ring:

Isolated knowledge-bases on any very specific topic may be open for a very thin segment of students, but combining all the data-bases into one server may make the data-base very very large. The best compromise between speed of access and width of coverage of the topics, is to arrange the knowledge-servers on a local-area network. The knowledge-servers may be arranged using a ring or hierarchical bus architecture as in fig 163 or 164.

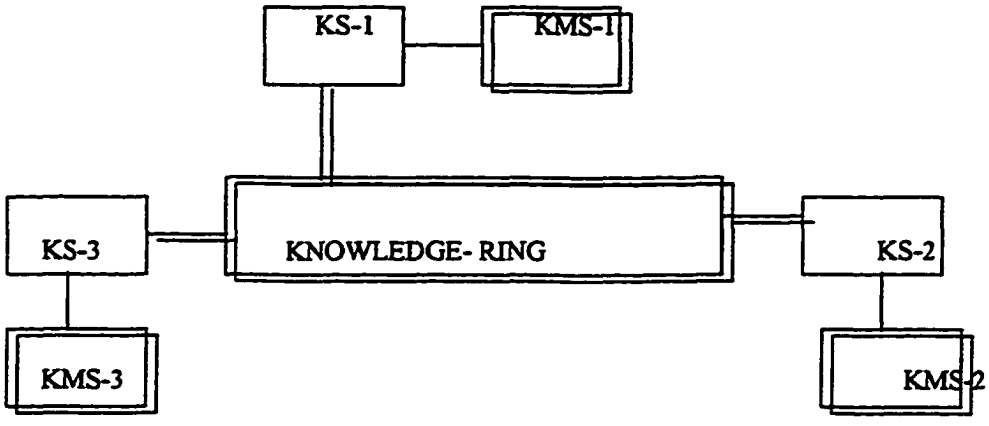


Figure 163:Arrangement of a knowledge-ring

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

There are many other possible topologies of interconnectivity among knowledge servers. If the knowledge servers are networked, then the knowledge-servers will need to be assigned logical addresses (although the physical addresses may be anything) according to the classification of Dewey decimal system or Library of Congress on the subject itself.

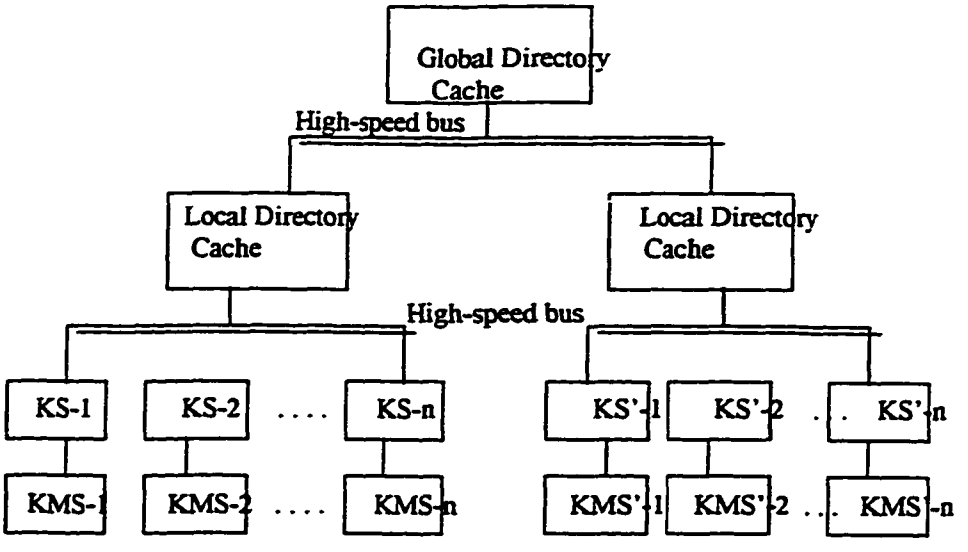


Figure 164:A hierarchical bus arrangement of KSs

4.5.4.2.2 Knowledge Management System (KMS):

The knowledge-content of the servers will continually need to be updated, rearranged etc.. These may be done through some automated feed or using expert people working with an expert shell. In the long run, probably the updates will be automated. The classifying task will done only once or it will be done using automated statistical classification schemes. The classified documents will be fed electronically and the same text may be classified into multiple classifications.

4.5.4.3 New elements in the network:

4.5.4.3.1 Knowledge Query Point (KQP):

This is the first software interface between the CN and NBIS. This element can exist on the high-speed data network acting as the network access point on the NBIS. This processor equipment serves to

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

comprehend the user's query interactively, determines the type of solicitation and subject matter and also acts as a gateway to the NBIS generating the transaction on NBIS. The solicitation can be any of the what, how, when, where, why etc. or combinations. Each such code results in an operation code. In addition, the subject matter generates a subject code according to the subject classification scheme (may be Dewey classification scheme or Library of Congress scheme, Asian Library system or Princeton classification system etc.). The user's query thus results in single-step or multi-step network instruction with operation and operand codes.

4.5.4.3.2 Knowledge Control Point (KCP):

This processor element converts the knowledge identifier (subject matter code) into logical address of the knowledge servers. Depending on the translated addresses, it may mean broadcasting the assembled code to multiple KSs or routing the request to a single KS on the IN. The address generated should be hierarchical in nature, such that the first part of the address identifies the respective KS, the second part specifies a range of search (the start point & end point for the limit of search). The address translation table (from subject code to logical address of KSs) can be implemented on the RAM so that the translation completes fast.

4.5.4.3.3 Knowledge Logic Interpreter (KLI):

These small software units of logic code exists in the KCP element and is responsible for sending the binary request code to the knowledge server and vice versa. The binary code can be executable or it may be references to the micro-code which is already loaded in the knowledge-server. The KLI is responsible at the KR (knowledge ring)-end of the IN to bring down the responses. The response packet-data may come through a common data network using X.25 or BX.25 etc. protocol to the CN.

4.5.4.3.4 Knowledge Transfer Point (KTP):

These processing elements are equivalents of STPs in IN, providing switching from SCP to the respective KSs.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.5.5 NBIS network protocol:

Although the protocol for KCP-KTP-KR interaction on the backbone looks similar to the TCAP transaction, but CCS7 protocol will not be enough to describe it because the response packets may come down using X.25 protocol. A superset of OSI protocol and CCS7 protocol may become redundant. If the KCP-KR protocol can be CCS7 protocol than an STP can be used for KTP. Safest will be to design a protocol for the knowledge transaction on the NBIS network.

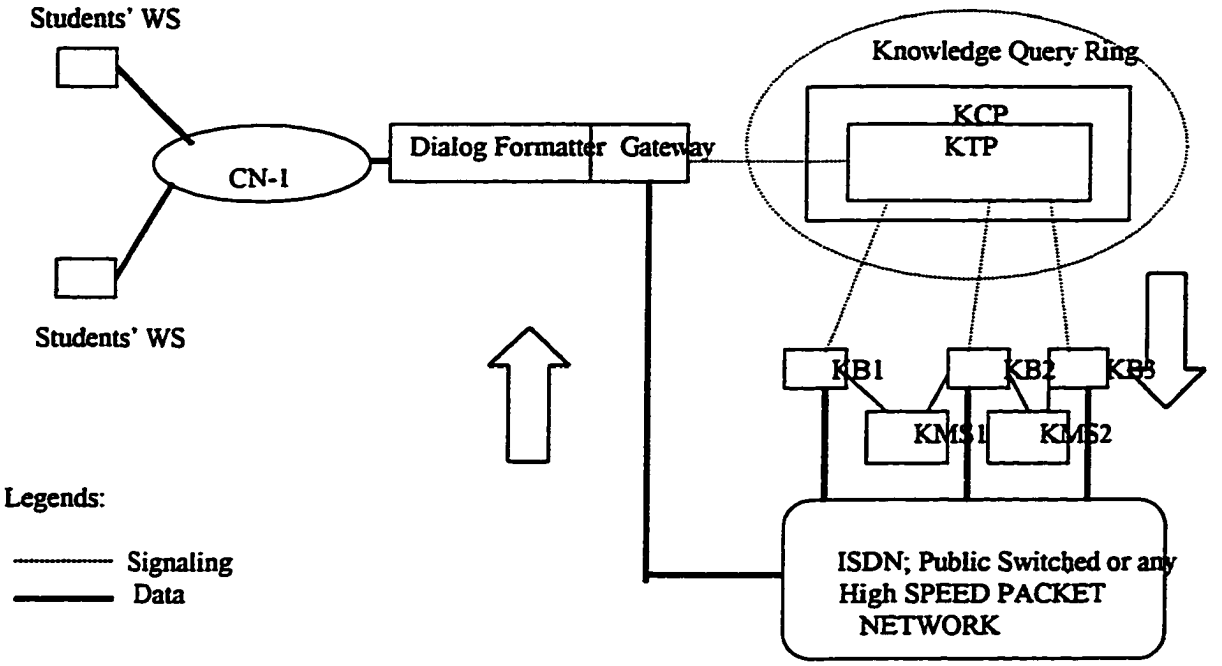


Figure 165:NBIS transaction flow diagram

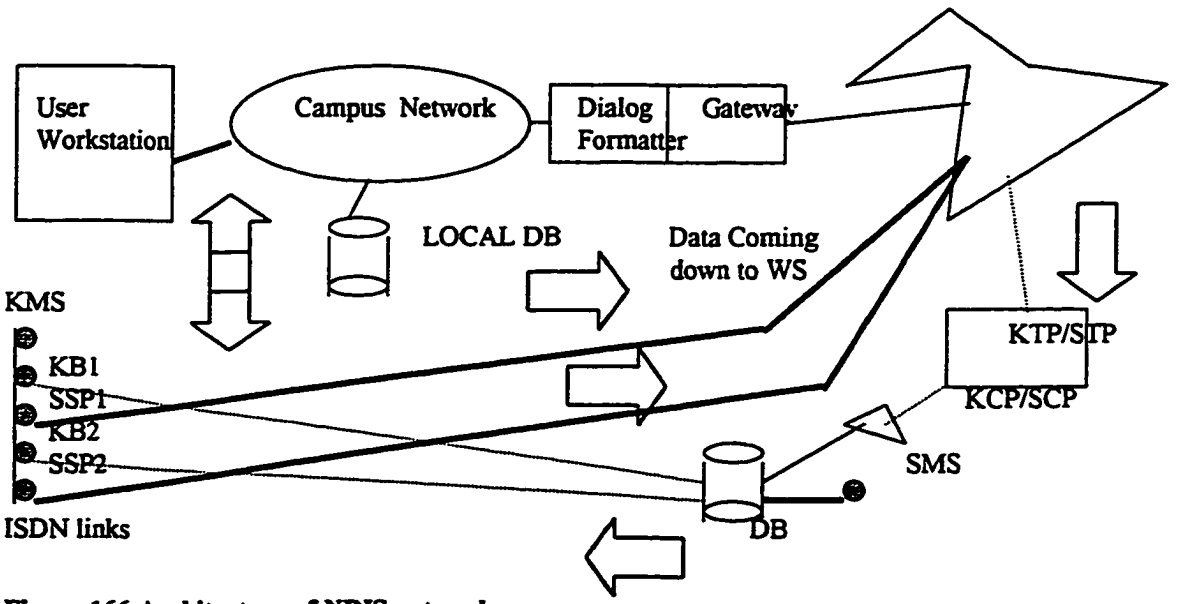


Figure 166:Architecture of NBIS network

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.5.6 Architecture of NBIS:

For this NBIS network, the architecture is illustrated in fig 166 and the transaction flow diagram is in fig 165. Fig 166 shows all the components of the NBIS network and how they are connected. Fig 165 shows how the transaction control flows from element to element.

Users distributed over the CNs enter their queries which are routed out through the gateways to the KQPs sitting on the packet data network (or it may be a circuit switched network providing a switched connection over a long period of time). A packet data network suits better than a circuit switched network for its better speed performance. The KQP may be located either on the public data network or on the backbone network. KQP generates operation, operand combination and also generates the other details of the request. KCP is part of the backbone network which generates the destination address and the binary code. KTP switches the request to the KSs using the high speed backbone network. This upward path is shown with the continuous lines. The KSs and the KMSs at various locations may be interconnected.

The return path of the response can be independently a separate path. The SLI at the nearest SCP may generate a signaling request for the local SSP (local to the KS) to set up a VC through ISDN line or any line on the public data network. A data formatter at the CN will receive the response & display to the user. Here exists a difference in functionality of KTP and STP because KTP needs to allocate a one-way path to the KS than a bi-directional path.

4.5.7 Intelligent functions of NBIS:

4.5.7.1 Supplementary knowledge:

As all the users queries can be stored at the KCP (with their semantics), a meta-logic process can be used to detect any contradiction within the progress of the logic in a few subsequent queries. Also if there is any subject area where the conversation flow did not take place subsequently, it can detect them and notify the user if she is ill-informed of any subject.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.5.7.2 Complementary knowledge:

As the queries continue, the query processor can build a specification tree using the conceptual graph. The query processor can even extrapolate from this and keep an account of possible growth of such graphs. Although doing this endlessly will induce infinite possibilities, but extrapolating the graph by adding one or two additional relationship ahead may provide a better concurrent processing model and also it may show the user which queries she did not pose.

4.5.7.3 Deadlock salvaging:

There are situations in user interaction, when user hits a dead-end and further queries resume in a different direction. As the system already knows the profile of the user (i.e. the areas of the user's research), it can apply some forward chaining and backward chaining to find any logical path connecting semantics at the deadlocks and the areas of user's interest. If such path exists, it can advise the user not to take different turn, but continue queries along the suggested line.

As a whole, NBIS can be potentially an effective source of knowledge for serious researchers engaged in long-hour interaction for knowledge retrieval in diverse areas and interdisciplinary topics.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A

Volume II

Distributed Intelligent Information Network

by

Apurba K Saha

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

1999

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5. Distributed Intelligent Information Network-Information Retrieval (DIIN-IR)

5.1 Introduction

A network-supported knowledge-based text retrieval framework has been presented here using the paradigms of object-oriented representations and Artificial intelligence and this will be used to answer queries for text retrieval from the user. Although this is not a question-answer system, but a modified version of this design can be used for such a purpose also. This framework is based upon a semantic model which is a popular IR model today and it is based upon two major principles: 1) intelligent query analysis and 2) knowledge-based information retrieval.

This model is distinguished for its semantic knowledge in query evaluation instead of pure literal query matching. This semantic model will process its semantic knowledge in three various ways: 1) classification (assigning objects to classes) of a query term or a retrieved item, 2) aggregation (referring to a composite class by a single name), 3) generalization (forming a hierarchy of classes). This proposed semantic model uses its own set of rules (from its knowledge-base) of properties for applying these operations. One apparent limitation of semantic model is that it restricts the retrieval to selection operation only [14]. This model suggested here overcomes this limitation by supporting users with optimal queries and supporting selection, ranking [18], indexing.

5.2 Basic Principle:

In the context of knowledge-base processing, each retrievable document is an object and a set of objects sharing some important common broad subject (based on general concept) forms an object class. As the broad concepts can be arranged in a hierarchy of abstraction, any major concept can be located in this tree of subject classes. An interdisciplinary topic may represent a class adhering to multiple inheritance thus the class hierarchy not strictly following a tree structure. A generalized knowledge-base can be set up among the concept classes of the domains representing relationships of different kinds among the concepts.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

One common example of such relationship is the "isa" relationship which indicates hierarchy among the concepts. But there are other relationships also as opposite, synonym, similar, cause, consequence, affect, precedes, follows, concomitant, mutually exclusive etc..

From parsing the user's query, the keywords are identified and from those an initial judgment is made about the subject matter of the query, the major and minor (meaning the least general and most specific) topics are identified and the query can be bound to one or more nodes in the subject class hierarchy. For this purpose, the component concepts are extracted from the query and then through generalization[15] and use of concept distances [19], the broad subjects are identified. In addition, the history of the past queries of each user may be made available from the user's profile for assistance in analyzing the main concepts in the given query. The user's profile contains information about the topics of the user's research interest, a history of past queries & the past subjects etc.. Using an initial world-knowledge and metarule-set, the main concepts underlying the user's query are extracted.

Generalizing the component concepts of the given query and the pre-established queries in the historical knowledge for the user, the keywords in the query are identified in their decreasing sequence of importance, the key phrases and the complex conceptual structures are identified that are present in the query and are converted to elementary knowledge structures using a model of thematic roles details of which are described later. The general topics are maintained as a knowledge-base also containing a key item which is the Dewey decimal subject classification index (alternately it can be Library of Congress classification index).

The information database used in an heterogeneous one which is distributed over many locations on the public network. Each database can have its own independent data-model, database algorithms etc. but each supports a common query interface. Such interface supports queries coming in with the specified semantic concepts and the associated arguments (represented as structures). Based upon the subject matter, types of information sought, depth of specificity of the query etc. a set of nodes have to be determined that need to be searched for answering the query. This set of nodes is built from the subject index. Initially a

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

small preferred subset (for minimized cost) is used, but if sufficient answers are not received, the set is expanded incrementally.

During the query answering phase, the model of an open DAI model is used for distributed interacting using a blackboard. This blackboard will be the object carrying the subqueries, messages for each node, the set of node names, the retrieval set etc. The spawned processes at individual nodes as a result of the subqueries will be the representing agents or brokers. The agents will negotiate about who will put what results on the blackboard, if more nodes should be activated for generating the answer, what are the broader topics and alternative topics etc.. The answer set will be re-ranked according to the relevance level for the documents. The details of the architecture will be explained later.

5.3 Software Architecture:

All the major software components involved in query-processing are explained below component after component. Fig 167 displays all the software layers involved in the system.

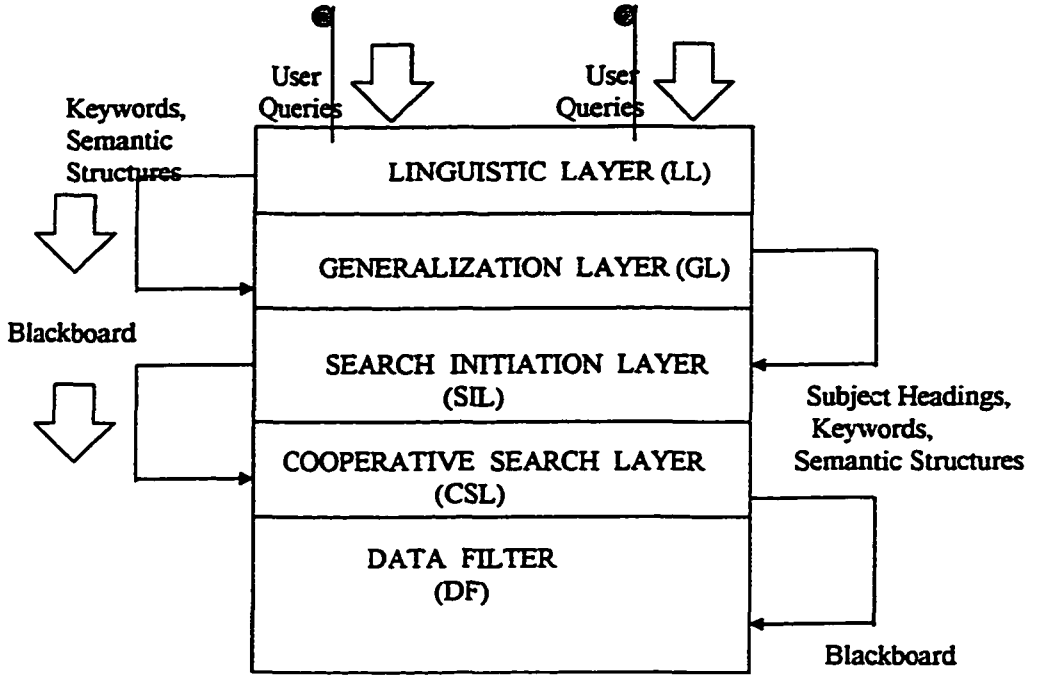


Figure 167:Layers of Software components

5.3.1 Linguistic Processing layer (LL):

(Semantic model and thematic roles)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In NLP, traditional IR systems ignore words as *when, at, between, for* etc. in a user query as empty although they carry important semantic information in the sense that they can be part of meaningful abstractions[17]. For example, *when* means time; *between* means duration or range; *for* means cause etc.. These thematic informations may combine with the attribute information to represent a query better. For simplicity, the thematic model can be a case-based model containing a finite set of thematic roles along with a world knowledge for thematic role analysis. Certain user words as verbs, prepositions etc. will trigger thematic role analysis. For example, *Fred knocked the door*. Here *Fred*'s role is the agent, *knocked* is the event and the *door* is the object. Thus the sentence can be tied to a plane of abstraction as agent-event-object and such representation is more useful because semantic search permits literally inexact matches. Using an object-oriented model, we can represent two classes of objects as 1)Thematic object, 2) Attribute object. Attribute objects are attributes of another entity or action. The thematic roles can be summarized as in table 5-1. For example to say, *how much does the table weigh?* results in an attribute frame as

Table 5-1: Basic thematic roles

Thematic Roles	Description
AGENT	Entity that causes an event or an action.
EVENT	Action involved with a verb.
OBJECT	Entity affected by an event.
LOCATION	Result of where question for an event.
TIME	Result of when question for an event.
REASON	Cause of an event.

Table.weight	?
Attribute	value

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

This semantics analysis-phase will be placed within the parser using a large world knowledge base with supporting meta-rules. The parser will also have a parts-of-speech tagger that will produce query in a simple form using the main keywords. The most important parts will be the proper nouns, the verbs and some other keywords. If there is any ambiguity in linguistic processing, the system may ask back the user for clarification. Finally this phase will result in producing the query using the keywords and the semantic structures as explained above. An example of the series of keyword concepts may be as :

(Computer-vision, 2) (Scene-Analysis, 5) (Graphic 3-D analysis, 7).

Where 2, 5, 7 are some weights assigned.

5.3.2 Generalization Layer:

This phase will apply the principles of generalization trying to reach the subject name so that the indexes for the database nodes can be determined. For this purpose, it will search the subject class hierarchy graph trying to match the keywords. Also it will measure the distances between the subjects of the past queries and the current subject confirming that the distance is small. If the distance is large, it will assume the current query to be a point of discontinuity and then the past queries will not be included in computing the modified query. The distance between two subjects is the measure of similarity among them. If there is a subject-concepts matrix formed which may appear as in fig 168.

Subjects S =

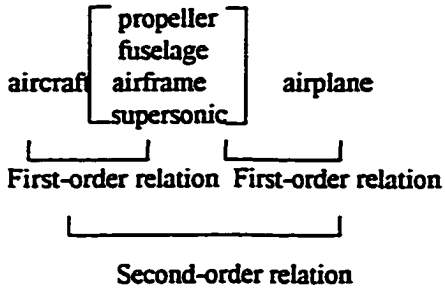
	Basic Concepts A	B	C	D	E	F
1	1	0	0	1	0	0
2	1	1	0	1	0	0
3	0	0	1	1	1	1
4	1	0	0	1	1	0
5	1	1	1	0	0	0

Figure 168:Subject-concepts matrix (S)

The subject-subject similarity matrix $S.S^T$ specify the first-order associations between the subjects. If any value is above a cutoff, then the distance may be 1 else 0. For the subjects, where the similarity matrix gives us 0, the second-order associations $(S.S^T)^2$ is computed and the cutoff is applied. The ones that have value above cutoff, the distance is given a value 2 else 0. Again for the subjects that have distance zero,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

again the third order associations are computed and the distance of 3 units are picked. In this way, one can compute distances between subjects. An example is the following:



Of course, computing this large matrices can be too highly complex. But what this means, is essentially this: if the two subjects are sub_1, and sub_2, then in the subject relationship knowledge base we need to look for a relationship between sub_1 and sub_2 for a direct relationship(distance 1), for an indirect relationship (or distance of 2) ,we need to find if there are two relationships as R_1 and R_2 such that sub_1 R_1 sub_i and sub_i R_2 sub_2. For the third-order associations, similarly one has to look for relationships R_1, R_2, R_3 so that sub_1 R_1 sub_i R_2 sub_j R_3 sub_2, for any i and j.

If the distances are small, then it will apply a set of generalization rules (rules of implication) trying to merge the series of subjects into a smaller and more general list which exist in the subject class hierarchy. The system may need to use forward-chaining and backward chaining involving heuristics trying to reach the subject names although it appears to be a bottom-up search trying to reach the closest subject from the keywords. Once this is done, the extra terms can be dropped of the list. As an example of generalization,

D1: (Computer-vision, 2) (Scene-analysis, 5) (Graphic 3-D Analysis, 7) {User Query}

D2: (Scene-analysis, 3) (Stochastic process, 4) {past history}

Using some rules, Computer-vision ⇒ Scene-analysis
Computer-Graphics ⇒ Graphic 3-D Analysis

Combining these two examples, the result subject becomes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(Computer-vision, 2) (Computer-Graphics, 7) (stochastic process, 4). The first two are in the subject headings whereas the third is not. So the subject list will be

Computer-vision or Computer-Graphics.

This subject list is passed down to the next layer along with the semantic representation of the query. The semantic representation of the query is comprised of 1) list of the underlying concepts in the query, its weights, its semantic roles in the context (or the sentence), its other attributes, 2) list of the semantic concepts embedded in the phrases and their weights etc..

5.3.3 Search Initiator Layer (SIL):

This layer of software is responsible for generating the required data, the process initiation and initializing other data structures that will be required throughout the processing of the given query. Once the subject-list is obtained, the database-nodes are determined from a cross-reference. The knowledge control point (KCP) possesses the knowledge containing the cross-reference of the node list where the query can be best run. The subject-nodes dictionary contains the list of hosts which carries good amount of information on the subjects specified in the subject-list computed before. This dictionary specifies upto the furthest details all the subjects and the major topics that each node (or host) carries information on. Searching this dictionary for the subject list and the major concepts list (but not by the semantic structures) corresponding to the query a preferred list of nodes is determined. The entries in the subject-node dictionary may simply look as in fig 169. To refine the logic further, more constraints can be added into consideration as network cost involved in carrying the data, host computer-charges etc..

Subject index	Concept	Node-name	Node-rating	No of volumes	Last year of Update
502.6.183.4	European War	U_Minnesota	3	450	1998
502.6.183.5	Influenza	New_York_PubLib	4	200	1997

Figure 169:Subject-node dictionary

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

At this point an instance of the *blackboard* class is created and the compiled query, the list of nodes, the list of preferred nodes are written on the blackboard. The blackboard moves from one site to another of all the preferred nodes and the queries are self-triggered at each of the preferred nodes. Effectively, the system spawns many knowledge sources (KSs) which function in parallel.

A KS is an intelligent agent that has the knowledge of its own aspect of the problem domain and is capable of solving the problem in its own domain. This KS method which may be called a blackboard system model was first used to develop an expert system for speech-processing.

In such a blackboard model, there is no specified central control component. A KS can only receive information from the blackboard and put its own results also on the blackboard. In other words, the system coordination and communication are solely realized by the information exchange on the blackboard. The blackboard provides some kind of communication channels among the knowledge sources. The constitution of a blackboard is shown in fig 170.

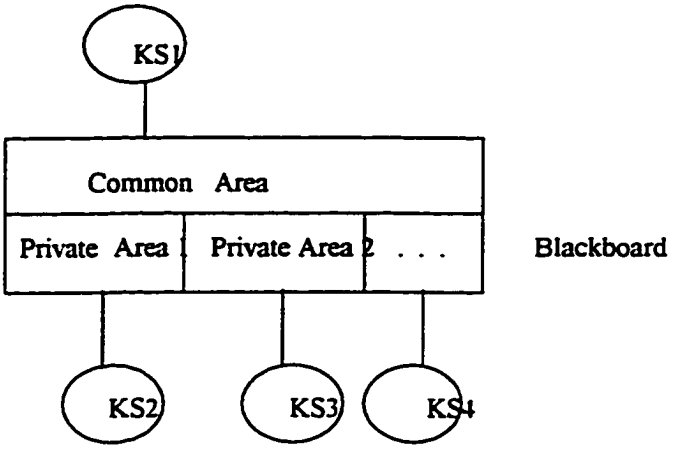


Figure 170:Internal organization of a blackboard

The blackboard is divided into areas among which there is a common area and a few private areas. The number of private areas depend on the number of KSs. Each KS can write or read its own private area and read or write into the common area. The common area works as an announcement board visible to all KSs.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.4 Query Relaxation:

When users' queries do not obtain any result, the *relaxation* technique enables the searcher to find other answers related to the original query. The relationships are defined through *taxonomy clauses* , which are first-order expressions about semantic relationships among predicates and constants in the deductive database. The taxonomy clauses are used to relax predicates, constants and variable dependencies in the users' queries.

For simplicity let us say that the database is made of first-order clauses and rules. A term is a constant, a variable or a function over constants, variables and functions and an atom is a predicate with terms in the arguments as in p(X, f(X)) where X is a variable and f is a function. A literal is a positive or negative atom. The syntactic information is made of *intensional database* (IDB) and the *extensional database* (EDB). The IDB is a set of clauses or rules of the form $A \leftarrow B_1, B_2, B_3, \ldots B_n$ where $n > 0$ and A and each B_i is an atom. The EDB is a set of clauses or facts of the form $A \leftarrow$, where A is a ground atom. Direct answers to database queries which are clauses of the form $\leftarrow B_1, B_2, \ldots, B_n$. Are found by applying resolution on the query, IDB, EDB clause to produce a search tree. The root node of the search tree is the query clause; each node in the tree is produced by applying an IDB rule to the node above.

General principle: Generalizing a query in order to capture neighboring information is a means to obtain possibly relevant information. A query can be relaxed in at least three ways: (1) rewriting a predicate with a more general predicate; (2) rewriting a constant term with a more general constant term; (3) breaking a join dependency across literals in the query. The first two relaxations can be achieved in a general manner using *taxonomy clauses* that define hierarchical type relationships between predicates and constants. For example, the following clauses define relationships between the predicates author_In_Field, highly_Overlapping_Fields, isa_Field_Within, wrote_A_Book and wrote_A_Paper.

T1: author_In_Field (Authorname, Fieldname, Title) ← highly_Overlapping_Fields (Fieldname, A),
wrote_A_Book (Authorname, A, Title).

T2: author_In_Field (Authorname, Fieldname, Title) ← highly_Overlapping_Fields(Fieldname, A),

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

wrote_A_Paper (Authorname, A, Title).

T3: author_In_Field (Authorname, Fieldname, Title) ← isa_Field_Within (A, Fieldname),

wrote_A_Book (Authorname, A, Title).

T4: author_In_Field (Authorname, Fieldname, Title) ← isa_Field_Within (A, Fieldname),

wrote_A_Paper (Authorname, A, Title).

The following query Q1 contains a request to verify if Roger Schank is an author in Computer Science, it can be relaxed to produce an alternative query Q1':

Q1: ← author_In_Field ('Roger Schank', 'Computer Science', 'The Cognitive Computer').

Q1':← isa_Field_Within ('Artificial Intelligence', 'Computer Science'), wrote_A_Book ('Roger Schank', 'Artificial Intelligence', 'The Cognitive Computer').

After the relaxation step, resolution methods can be used to find the answers.

5.4.1 Formal Definition

The taxonomy clauses in the database as shown in the examples above provide the basis for relaxation. They may be clauses that are already present in the database or they may be a separate set of clauses provided as semantic information about the database. But either way, they describe an implicit taxonomy. For example, T1, T2, T3 and T4 describe a taxonomy in which wrote_A_Book and wrote_A_Paper are two modes of authoring in a field.

For each taxonomy clause in the database, an additional *reciprocal clause* is added to the database. The body atom in a taxonomy clause that appears in the hierarchy is called a key atom. The head of a reciprocal clause contains the key atom of a taxonomy clause. The head atom in the taxonomy clause appears in the body of the reciprocal clause. For a taxonomy clause of the form H← B1, . . . , Bn, K, where

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

K is the key atom, the reciprocal clause is relax(K) ← B1, B2, . . . Bn, H. For T1, T2, T3 and T4 above, the reciprocal clauses are the following:

R1: relax (wrote_A_Book (Authorname, A, Title)) ← highly_Overlapping_Fields (Fieldname, A),
author_In_Field (Authorname, Fieldname, Title).

R2: relax(wrote_A_Book (Authorname2, B, Title2) ← isa_Field_Within (B, Fieldname2),
author_In_Field (Authorname2, Fieldname2, Title2).

R3: relax(wrote_A_Paper (Authorname3, C, Title3)) ← highly_Overlapping_Fields (Fieldname3, C),
author_In_Field (Authorname3, Fieldname3, Title3).

R4: relax(wrote_A_Paper (Authorname4, D, Title4)) ← isa_Field_Within (D, Fieldname4),
author_In_Field (Authorname4, Fieldname4, Title4).

Taxonomy clauses capture type relations over constants as well as over predicates. For example, the following taxonomy clauses represent some of the taxonomy relationships in fig 171.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

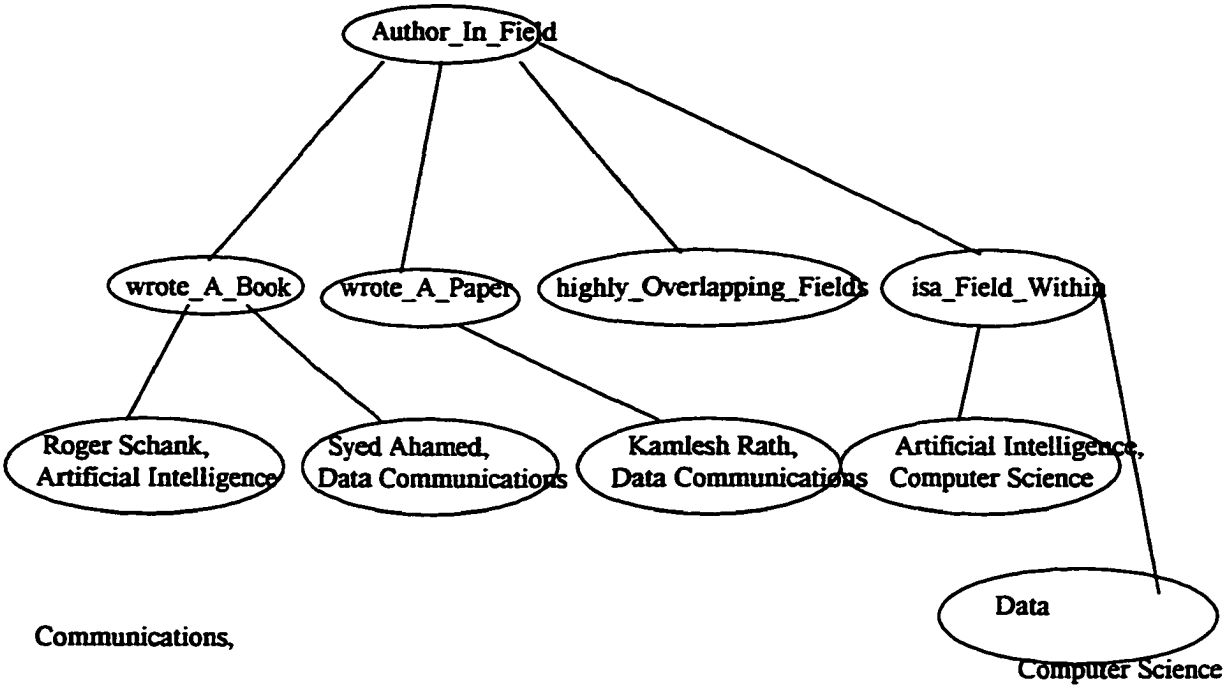


Figure 171:Type Taxonomy for Author in fields

T5: wrote_A_Book (A, B, C) ← A unifies 'Roger Schank', B unifies 'Artificial Intelligence', C unifies with 'The Cognitive Computer'.
T6: wrote_A_Book (D, E, F) ← D unifies 'Syed Ahamed', E unifies 'Data Communications', F unifies with 'Data Communications'.
T7: wrote_A_Paper (G, H, I) ← G unifies with 'Kamlesh Rath', H unifies with 'Data Communications', I unifies with 'Digital Video Networks'.
T8:isa_Field_Within(J, K) ← J unifies with 'Artificial Intelligence', K unifies with 'Computer Science'.
T9:isa_Field_Within (L, M) ← L unifies with 'Data Communications', M unifies with 'Computer Science'.

The reciprocal clause for T9 is the following:

R9: relax (M unifies with 'Computer Science') ← L unifies with 'Data Communications', isa_Field_Within (L, M).

Prior to relaxation, a query is *variable substituted* so that the constants are moved out into separate atoms. Variable substitution replaces constants and repeated variables in a query with unique variables. Then it equates the original constants and variables to the new variables. For relaxation, the equality expression uses the predicate *unifies*. For example, the variable substituted form of Q2 is Q2':

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Q2: ← wrote_A_Book ('Syed Ahamed', 'Data Communications', 'Data Communications').

Q2': ← wrote_A_Book (A, B),
A unifies 'Syed Ahamed',
B unifies 'Data Communications',
C unifies 'Data Communications'.

Normally the following meta-interpreter can be described to perform deduction together with relaxation to perform deduction together with relaxation using reciprocal clauses:

```
relax_solve (As) ← relaxing (As, Bs),
                   solve (Bs).

relaxing ([A|As], [B|Bs]) ← relaxing (As, Bs).

relaxing ([A|As], Bs) ← clause (relax (A), Cs),
                        relaxing (Cs, Ds),
                        relaxing (As, Es),
                        append (Ds, Es, Bs).
relaxing (□, □).
```

The second rule for the predicate *relaxing* calls the reciprocal clauses through the predicate *relax*. The predicate *solve* may perform deduction or apply cooperative answering methods to the query solving process.

5.4.2 User constraints

The user's restrictions on a database can be modeled as a set of constraints, called user constraints, and then used to alter a user's query. The query modification is performed by means of semantic query optimization. Here before offering an example, I assume another predicate relation called joint_Author defined as:

```
joint_Author (Author1, Author2, Fieldname, Title) ← author_In_Field (Author1, Fieldname, Title),
                                                    author_In_Field (Author2, Fieldname, Title).
```

5.4.2.1 An Example

Consider a query to a database as : who was the co-author of the co-author of 'B. W. Stuck' in the field of networks?

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

So,

Q3: ← joint_Author (Author1, Author2, 'Networks', Title), joint_Author (Author2, 'B.W. Stuck', 'Networks', 'Network Performance Primer').

And a user constraint exists as 'Ronald Malcolm is not considered a valid co-author'

U1: ← joint_Author (Author1, Author2, Fieldname, Title), Author1='Ronald Malcolm'

U2: ← joint_Author (Author1, Author2, Fieldname, Title), Author2='Ronald Malcolm'

with the following database information:

I1: joint_Author (Author1, Author2, Fieldname, Title) ← author_In_Field (Author1, Fieldname, Title), author_In_Field (Author2, Fieldname, Title).

I2: author_In_Field (Authorname, Fieldname, Title) ← highly_Overlapping_Fields (Fieldname, A), wrote_A_Book (Authorname, A, Title).

I3: author_In_Field (Authorname, Fieldname, Title) ← highly_Overlapping_Fields(Fieldname, A), wrote_A_Paper (Authorname, A, Title).

I4: author_In_Field (Authorname, Fieldname, Title) ← isa_Field_Within (A, Fieldname), wrote_A_Book (Authorname, A, Title).

I5: author_In_Field (Authorname, Fieldname, Title) ← isa_Field_Within (A, Fieldname), wrote_A_Paper (Authorname, A, Title).

E6: wrote_A_Book ('Roger Schank', 'Artificial Intelligence', 'The Cognitive Computer').

E7: wrote_A_Book ('Syed Ahamed', 'Data Communications', 'Data Communications').

E8: wrote_A_Book ('B. W. Stuck', 'Networks', 'Network Performance Primer').

E9: wrote_A_Book ('E. Arthurs', 'Networks', 'Network Performance Primer').

E10: wrote_A_Book ('E. Arthurs', 'Networks', 'Heterogeneous networks').

E11: wrote_A_Book ('Ronald Malcolm', 'Networks', 'Heterogeneous networks').

E12: wrote_A_Paper ('Kamlesh Rath', 'Data Communications', 'Digital Video Networks').

E13:isa_Field_Within('Artificial Intelligence', 'Computer Science').

E14:isa_Field_Within ('Data Communications', 'Computer Science').

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

E15:isa_Field_Within('Networks', 'Computer Science').

E16:isa_Field_Within('Artificial Intelligence', 'Artificial Intelligence').

E17:isa_Field_Within ('Data Communications', 'Data Communications').

E18:isa_Field_Within('Networks', 'Networks').

With only IDB ∪ EDB, the query can be answered as: *Ronald Malcolm is co-author to E. Arthurs who in turn is co-author to B. W. Stuck.* However, such an answer is not acceptable by the user because of constraints U1 and U2.

When query Q3 is expanded with the rules above, the modified query Q3′ appears as:

joint_Author (Author1, Author2, 'Networks', Title), joint_Author (Author2, 'B.W. Stuck', 'Networks', 'Network Performance Primer')

author_In_Field (Author1, 'Networks', Title), author_In_Field (Author2, 'Networks', Title),

author_In_Field (Author2, 'Networks', 'Network Performance Primer'),

author_In_Field ('B. W. Stuck', 'Networks', 'Network Performance Primer'),

wrote_A_Book ('E. Arthurs', 'Networks', 'Network Performance Primer'),

wrote_A_Book ('E. Arthurs', 'Networks', 'Heterogeneous networks'),

wrote_A_Book ('Ronald Malcolm', 'Networks', 'Heterogeneous networks'),

Author1 ≠ Author2,

Author2 ≠ 'B.W. Stuck'.

Merging Q3′ with U1 and U2 produces the following query Q3″ as:

Q3″ :

joint_Author (Author1, Author2, 'Networks', Title), joint_Author (Author2, 'B.W. Stuck', 'Networks', 'Network Performance Primer')

author_In_Field (Author1, 'Networks', Title), author_In_Field (Author2, 'Networks', Title),

author_In_Field (Author2, 'Networks', 'Network Performance Primer'),

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

author_In_Field ('B. W. Stuck', 'Networks', 'Network Performance Primer'),

wrote_A_Book ('E. Arthurs', 'Networks', 'Network Performance Primer'),

wrote_A_Book ('E. Arthurs', 'Networks', 'Heterogeneous networks'),

wrote_A_Book ('Ronald Malcolm', 'Networks', 'Heterogeneous networks'),

Author1 ≠ Author2,

Author2 ≠ 'B.W. Stuck',

Author1 ≠ 'Ronald Malcolm',

Author2 ≠'Ronald Malcolm' .

Now this query Q3" fails because it is false. So the query expansion fails due to the user's constraints.

Definitions

Before going into some definitions, we assume that the notion of the user constraints, the extensional database and intensional database contain Horn clauses and are function free. The close world assumption is also assumed. The DB is a set of IDB rules and EDB facts. Let L be the language of DB and let E be the following set of evaluable predicates: {=, <, >, ≥, ≤, ≠, is with +, -, *, ÷}.

User Constraint

A user constraint is a clause of the form ← P1, P2, . . . ,Pm, E1, E2, . . .,En, where each Pi is a literal in the database language L and each Ej is an evaluable atom with a predicate in E.

In order to restrict the answer set of a query with a set of user constraints, the user constraints are semantically compiled into the query.

User-constrained rule

A user-constrained rule is defined to be of the form

$A \leftarrow B_1, \ldots, B_m, \{R_1, R_2, \ldots, R_n\}$.

Where A is a procedure head of the rule, each literal B_i is either an extensional predicate or an evaluable predicate, and the R_js are residues obtained from the rule $A \leftarrow B_1, B_2, \ldots, B_m$ and some set of user constraints.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

User-constrained database

Let DBS be a semantically constrained database with IDB rules, IDB. Let UC be a set of user constraints in the language of DBS. The user-constrained database of DBS, DBS', is the set of user-constrained rules obtained from IDB and UC.

If a user-constraint is semantically compiled into a query, then the answers for the new query reflect the restrictions of the user constraint.

Query answer

Let Q be a query, $\leftarrow Q_1 \ldots Q_n$, where each Q_i is a literal.

Let $\{\theta_1, \theta_2, \ldots \theta_m\}$ be a set of substitutions such that for each θ_i, DB $\vdash$ $(Q_1, Q_2, \ldots, Q_n)\ \theta_i$.

Then each θ_i is an answer for Q.

There are some distinctions among integrity constraints and user constraints. An integrity constant, IC always holds true in a database - that is DB $\cup$ IC is consistent. No facts nor rules in the database violates the IC rules. But this is not necessarily true of a user constraint. User constraints carry different meanings since we wish to use user constraints to check answer substitutions that have been obtained for a query. This means the answers to the query satisfies the user's constraint, but the rest of the database may not satisfy the user's constraints.

Intuitively, we can check each answer substitution θ by applying it to each variable substituted user constraint, U, that partially contains the query with the one-way substitution σ. If the database contains any of the instantiated user constraints, that is DB $U\sigma\theta$, then the answer substitution is acceptable. If the database does not entail the instantiated user constraint, then the answer substitution is considered to *violate* the user constraint and is rejected.

The following theorem states this formally.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Theorem 1 Let Q be a flattened query $\leftarrow Q_1 \ldots Q_n$, where each Q_i is an EDB literal and let U be a flattened, variable substituted user constraint, $\leftarrow P_1, \ldots, P_m, E_1, E_2, \ldots E_r$, where each P_i is an EDB literal and each E_i is an evaluable atom. Let S be a subset of atoms in $\{P_1, \ldots P_m\}$ that subsumes some subset of atoms in $\{Q_1, \ldots Q_n\}$ with substitution σ. Let C be the conjunction of P_is in $\{P_1, \ldots, P_m\}$ - S. Let Q' be a new query obtained by semantically compiling Q and U. Q' has the form:

$Q_1, Q_2, \ldots Q_n, \neg (C\sigma, E_1\sigma, \ldots, E_r\sigma)$.

Let $\{\theta_1, \ldots \theta_p\}$ be a set of answers for Q', that is, $DB \vdash (Q_1, \ldots, Q_n)\ \theta_i$, $1 \leq i \leq p$. Then $DB \vdash U\sigma\theta_i$ $1 \leq i \leq p$.

A proof of this theorem is available in [88].

A short example helps to clarify the relationship between an answer substitution and a user query constraint.

Consider the query Q that says, *who wrote the book 'Data Communications' ?* and a variable substituted user constraint U that says, *exclude the books of William Stallings.:*

So

Q4: ← wrote_A_Book(Authorname, Fieldname, 'Data Communications').

U: ← wrote_A_Book (AA, FF, TT), AA = 'William Stallings'.

Here U partially subsumes Q4 with the substitution σ = {Authorname/AA, Fieldname/FF, 'Data Communications'/TT}.

The semantically compiled query is the following:

Q4' : ← wrote_A_Book (Authorname, Fieldname, 'Data Communications'), Authorname ≠ 'William Stallings'.

Suppose the EDB has the following facts.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

wrote_A_Book ('Roger Schank', 'Artificial Intelligence', 'The Cognitive Computer').

wrote_A_Book ('Syed Ahamed', 'Data Communications', 'Data Communications').

wrote_A_Book ('Lawrence Reggie', 'Data Communications', 'Data Communications').

wrote_A_Book ('William Stallings', 'Data Communications', 'Data Communications').

wrote_A_Book ('B. W. Stuck', 'Networks', 'Network Performance Primer').

Then there are two answers for the query Q4', as θ_1 and θ_2:

θ_1 = {'Syed Ahamed'/Authorname, 'Data Communications'/Fieldname}.

θ_2 = {'Lawrence Reggie'/Authorname, 'Data Communications'/Fieldname}.

Each answer substitution can be applied to the user constraint to obtain two instantiated constraints. $U\sigma\theta_1$ is the following:

← wrote_A_Book ('Syed Ahamed', 'Data Communications', 'Data Communications'),

'Syed Ahamed' = 'William Stallings'.

and $U\sigma\theta_2$ is the following:

← wrote_A_Book ('Lawrence Reggie', 'Data Communications', 'Data Communications'),

'Lawrence Reggie' = 'William Stallings'.

In this case, for i=1 and i=2, DB$\vdash$ $U\sigma\theta_i$ trivially since the body of the user constraint evaluates to *false*, thus making the entire user constraint evaluate to *true*.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.4.3 Limiting Relaxation with User Constraints

Users may express their own concerns writing queries. But when a query is relaxed, these concerns may be dropped partially or fully. Here it is shown that the user constraints as described in prior section is an excellent way to separate properties that a user wishes to hold for every query from the properties that a user has assembled for a particular query. Consider the following query which asks for authors for books in the Fieldname 'Neural Networks' and also asks if this field is similar to 'Networks' or not:

Q5: ← wrote_A_Book (Authorname, 'Neural Networks', Title), similar('Neural Networks', 'Networks')

Variable substitution, relaxation, and deduction with R1, R3 and T1, T3 produce two new relaxed queries:

Q51': ← wrote_A_Book (X, Y, Z),
highly_Overlapping_Fields (Y, Y1),
highly_Overlapping_Fields (Y1, 'Neural Networks'),
similar ('Neural Networks', 'Networks').

Q52': ← wrote_A_Book (X, Y, Z),
isa_Field_Within (Y, Y1),
isa_Field_Within (Y1, 'Neural Networks'),
similar ('Neural Networks', 'Networks').

If the user's intent in the original query was only to make sure if the topic of the book is similar to 'Networks' or not, then the intent has been lost in the relaxed query. Possibly it may be better to allow the user to express this intent through a user constraint as *I want to know only those books that are similar to 'Networks'*.

U2: similar (X, 'Networks') ← wrote_A_Book (X, Y, Z).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Given the query:

Q6: ← wrote_A_Book (Authorname, 'Neural Networks', Title).

Semantic query optimization with U produces Q5 above. But relaxation of Q6 produces the following query:

Q61': ← wrote_A_Book (X, Y, Z),
highly_Overlapping_Fields (Y, Y1),
highly_Overlapping_Fields (Y1, 'Neural Networks').

Q62':← wrote_A_Book (X, Y, Z),
isa_Field_Within (Y, Y1),
isa_Field_Within (Y1, 'Neural Networks').

Applying the user constraint U2 to relaxed queries Q61' and Q62' produces the restricted queries that makes sure that the topics are similar to 'Networks' :

Q61'': wrote_A_Book (X, Y, Z),
highly_Overlapping_Fields (Y, Y1),
highly_Overlapping_Fields (Y1, 'Neural Networks'),
similar (X, 'Networks').

Q62'': wrote_A_Book (X, Y, Z),
isa_Field_Within (Y, Y1),

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

isa_Field_Within (Y1, 'Neural Networks'),

similar (X, 'Networks').

In another case, user constraints can be used to eliminate relaxations that the user would not want to see. For example, suppose a user expresses that he does not want books written by juvenile authors.

So U3: ← wrote_A_Book (X, Y, Z),

adolescent(X).

If the user asks about books in 'Computer Science' using query Q7, two relaxed queries using R1, R3 and T1, T3 would be Q71' and Q72'.

Q7: ← wrote_A_Book (X, 'Computer Science', Z).

Q71': ← wrote_A_Book (X, Y, Z),

highly_Overlapping_Fields (Y, A),

highly_Overlapping_Fields (A, 'Computer Science').

Q72': ← wrote_A_Book (X, Y, Z),

isa_Field_Within (Y, A),

isa_Field_Within (A, 'Computer Science').

The user constraint U2 applies to these relaxed queries to produce two new queries Q71" and Q72".Both the queries are eliminated because no EDB facts support adolescent (X).

Q71": ← wrote_A_Book (X, Y, Z),

highly_Overlapping_Fields (Y, A),

highly_Overlapping_Fields (A, 'Computer Science'),

¬adolescent(X).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Q72": ← wrote_A_Book (X, Y, Z),

isa_Field_Within (Y, A),

isa_Field_Within (A, 'Computer Science'),

¬adolescent(X).

So the query fails as the last clause can not be satisfied.

Among the two examples cited here, the first one illustrated that user constraints can be used to retain persistent properties separately and apply them into relaxed properties. The second example shows how user constraints can be used to express limiting information that can eliminate possible relaxations.

5.4.4 Heuristics for Ordering Relaxations

Integrity constraint rules can be used to correct misconceptions in user's queries. Even there are heuristic methods available for selecting which integrity constraints or false presuppositions to return to a user. This approach is described below.

5.4.4.1 Background:

Integrity constraints specify unchanging states of the database like *Every entity that is an author must have a high school degree in Denver* and *No entity is both an author and a publisher in California.* Cooperative systems have been designed in the past [89] using integrity constraints to provide alternative responses to users. If a user asks a query such as *Who are the people in California who are authors as well as publishers?* , the query should fail to answer. A literal answer is *none*. A better answer would contain the constraint information, *No one is both an author and publisher in California.* Alternatively, a user may ask, *who are the authors with high school degrees in Denver?* A cooperative and informative answer will tell the user, *All authors have a high school degree in Denver.* The system may go further on and list all the authors from Denver or it may pause momentarily asking if the user needs a full list of all the authors.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In both cases, the user's query has a misconception about the contents of the database, and the integrity constraint information corrects that misconception. In the former case the misconception causes the query to fail; in the latter case the misconception results in the query to cover extra search space to answer the query.

The cooperative answering system uses semantic query optimization techniques to identify interactions between integrity constraints and queries that demand the cooperative information. The system also searches for false presuppositions of failed queries. Letting a user know about a false presupposition is an alternative way of providing an informative answer.

A set of heuristic rules determines which pieces of cooperative information ought to be included when multiple pieces are available. In addition, the user also has an option to ask for more explanatory answers or more precise answers.

5.4.4.2 Ordering Relaxations

We order relaxations by means of a set of heuristics that inspect any misconceptions in a query that have been identified through query optimization methods. In the following heuristic rules, a misconception is considered to be a set of subgoals which ensures that a query will fail:

1. Relaxation of misconceptions is given priority over other relaxations.

2. When several misconceptions are available for relaxation, precedence levels are set up as follows:

   (a) Relaxation of a misconception detectable without an EDB search has priority over misconceptions detectable only with an EDB search.

   (b) Relaxation of a misconception discovered through constraints takes precedence over those discovered as false presuppositions.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(c) Relaxation of a misconception that can be detected with a single constraint has priority over those detectable only with several constraints.

3. When several misconceptions have the same relaxation priority, the misconception identified by the constraint with the deepest derivation is selected for relaxation. Otherwise, the misconceptions are randomly ordered.

To illustrate these rules for ordering relaxations, consider the following integrity constraints which say that only military personnel can be authors of books on martial arts and no book on martial arts is jointly authored if the first author is from Denver:

IC1 : ← author_In_Field (Author, 'Martial Arts', Title),
¬ military_personnel (Author).

IC2 : ← joint_Author(Author1, Author2, 'Martial Arts', Title),
from_Place (Author1, 'Denver').

IC3: ← joint_Author (Author1, Author2, 'Martial Arts', Title),
from_Place (Author2, 'Denver').

All the constraints apply to the query. Can Dr. Silva author a book on martial arts with Mr. Robin from Denver?

Q8: ← joint_Author ('Dr. Silva', 'Mr. Robin', 'Martial Arts', Title).
from_Place ('Mr. Robin', 'Denver').

IC1 would require an EDB search on military_personnel because IC1 will clear only military personnel. But to detect a violation requires a full search on the EDB for military_personnel. Same thing is true for

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

IC2 for the predicate from_Place. On the other hand, IC3 does not require such a full DB search. Q8 violates IC3 directly without needing to search anything else. Thus relaxation of the atoms joint_Author ('Dr. Silva', 'Mr. Robin', 'Martial Arts', Title) and from_Place ('Mr. Robin', 'Denver') would be preferred over relaxation of author_In_Field (Author, 'Martial Arts', Title).

If no constraint identifies a misconception, but several false presuppositions are available, the most general false presupposition is selected for relaxation. Consider a variable substituted query that violates neither of the above constraints:

Q9: ← author_In_Field (Authorname, Topic, Title),

Topic unifies 'Martial Arts', Title unifies 'Introduction to Martial Arts'.

Asking for authors to the book or article titled 'Introduction to Martial Arts' on topic 'Martial Arts'. It contains a presupposition that literature on 'Martial Arts' exist; it also contains the presupposition that literature on 'Martial Arts' titled as 'Introduction to Martial Arts' exist. If the database contains no information about such literature, the query will fail and both presuppositions will be false. According to the relaxation ordering rules, the atom corresponding to the more general false presupposition would be relaxed - Topic unifies 'Martial Arts'.

These heuristics for ordering the relaxations have several important properties. Rule 1 targets misconceptions to be relaxed first. The relaxation process formulates a new, corrected query for a user. The user then has the option to use the relaxed query or to pose an entirely new query.

Rules 2a, 2b and 2c make the query answering process more efficient. When misconceptions can be found without EDB search, they are posed as sites of relaxation before any other misconceptions are sought. Identifying misconceptions via integrity constraint violations is much less expensive than searching for least failed subqueries of a query. Thus, any search for false presuppositions should be delayed until after integrity constraint violations are relaxed.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Rule 3 deals with derived constraints. When constraints take the form of a rule, as in IC3, it may be possible to derive new constraints like IC5 from IC3 and IC4:

IC3: triangle_Joint_Authors (X, Y, Z, SubjectIndex) ← joint_Author(X, Y, SubjectIndex, Title),
joint_Author(Y, Z, SubjectIndex, Title1),
joint_Author(X, Z, SubjectIndex, Title2).

IC4: ← triangle_Joint_Authors (X, Y, Z, SubjectIndex),
Fieldname (SubjectIndex, 'Computer Science').

IC5: ← Fieldname (SubjectIndex, 'Computer Science'),
joint_Author(X, Y, SubjectIndex, Title),
joint_Author(Y, Z, SubjectIndex, Title1),
joint_Author(X, Z, SubjectIndex, Title2).

Here IC3 says that any set of three distinct joint authors as X, Y, Z having produced three literatures in a joint style such that X and Y produced one, Y an Z produced another and X and Z produced another literature - will be a triangle among joint-authors. Integrity constraint IC4 says that no triangle exists among joint authors in 'Computer Science'. IC5, derived from IC3 and IC4, says that no authors in the area of 'Computer Science' seem to have triangular relationship such that say authors, X and Y wrote a literature jointly, authors Y and Z wrote another literature jointly and authors X and Z wrote a different literature jointly.

If all three of these constraints were violated by some query, Rule 3 would give higher relaxation priority to the atoms involved in the violation of IC5, since IC5 would reflect the misconceptions identified by all three constraints.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The heuristic rules for ordering relaxation can be used in two ways. They can order a set of possible relaxations before the relaxations are presented to the user as alternatives, or they can be used to automatically select a relaxed query. Selecting between these two modes of interaction is an open problem. It requires testing of users' ongoing interactions with a system. The better choice may be sensitive to the domain of the database and the level of expertise of the user.

5.4.5 Processing Queries

To use user constraints to restrict relaxation and to order possible relaxations with the heuristic rules given above, a user's query is processed in the following steps:

1. A query Q is processed with deduction as usual or with cooperative answering methods as outlined before.

2. Additional relaxed queries are found as follows:
   (a) If the query succeeds but the user wants additional answers, relaxation is applied to find an ordered set of new related queries, $\{R_1, R_2, \ldots R_n\}$.
   (b) If the query violates integrity constraints, the heuristic rules as described above are applied to obtain an ordered list of relaxed queries, $\{R_1, R_2, \ldots R_n\}$.
   (c) If the query fails and has no integrity constraint violations, false presuppositions are sought and the rules as described above are applied to obtain an ordered list of relaxed queries, $\{R_1, R_2, \ldots R_n\}$.

3. The set of user constraints are applied to each new relaxed query to form $R_1', \ldots, R_m'$. Any R_i that violates a user constraint is eliminated.
4. The remaining relaxed queries are either presented to the user so that the user can decide what to do next, or the highest priority relaxed query is selected for relaxation.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.4.6 Comments

The process of query relaxation provides a user with a means to automatically identify new queries that are related to the user's original query. However, for large databases, many relaxations may be possible. The methods to control and restrict the relaxation process (using user constraints) as described here focus the relaxation process and make it more efficient. The interaction between the relaxation method, the user constraint method and the misconception identification method can be seen as a cooperative answering system. In general, a cooperative activity consists of one entity helping another entity to achieve one or a set of goals. In the context of knowledge-based systems, the goal of the user is to gain particular information by posing queries and receiving answers. The goal of the system is to answer not just the literal query but the intended query.

It is worthwhile to consider how these behaviors may apply in the context of two or more knowledge-bases exchanging information in order to achieve some goal. There can be some generalizations of the user/system interactions or intrasystem interactions. Some of the scenarios are discussed below:

- Consider two processors each with a separate knowledge-base. In a situation in which one processor sends a sub-query to another processor for solution, if integrity constraint violations are detected at the second processor, the misconception should be returned to the first processor. Whether the first processor should incorporate the misconception into the answer or use the information to "learn" that the second processor cannot answer queries that violate the IC in question, is an open question. Detection of integrity constraint violations becomes a means to determine whether or not a particular processor can provide an answer to a sub-query. This may be useful for guiding communication between processors or for building a more informative answer.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Suppose that a first processor sends a query to a second processor that is not available and that the second processor is the only one with the definite information to solve the query. If a third processor could solve a relaxed version of the query, then the overall system could at least provide related answers to the original query. Relaxation here becomes a means to redirect subqueries.

- Suppose that a first processor divides a query into a local query and a remote query and that the link to the remote processor is slow. The first processor could formulate a "user constraint" to eliminate the remote portion and then include in the answer to the theory of user constraints. User constraints becomes a means to return intermediate answers quickly and to characterize them so that the user knows the answer is incomplete.

In each of these scenarios, the critical point is that the cooperative behavior between two entities allows them to provide answers that they could not provide without cooperation.

Another common knowledge-base formalism used for relaxation and restriction is constraint satisfaction network. The constraint satisfaction problem (CSP) deals with the assignment of values to variables according to existing "constraints". Given n variables with their finite domains along with the admissible combinations of their values it is the goal to explicate all instantiations still possible resulting in a set of n-tuples which represents the globally consistent solution.

5.5 Distributed Knowledge-base representation using CSP

Knowledge-base formalisms for representing distributed knowledge-base are various; one special kind is a constraint network. The area outlined here is regarding intelligent change of a given problem description according to some interaction. This feature is important because one may not have handy the complete problem specification right from the beginning. This approach follows the principle just to change those parts which are really influenced directly by the modifications.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.5.1 Background

The constraint satisfaction problem (CSP) deals with the assignment of values to variables according to existing "constraints". Given n variables with their finite domains along with the admissible combinations of their values it is the goal to explicate all instantiations still possible resulting in a set of n-tuples which represents the globally consistent solution.

The following fig 172 symbolizes the structure of a CSP with four variables:

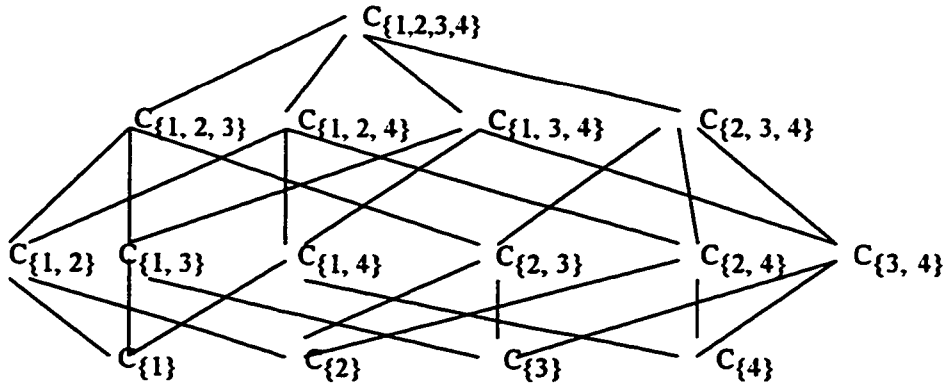


Figure 172: Structure of a CSP with four variables

The index set of one constraint enumerates the specific variables involved in the constraint declaration. E.g. : $C_{\{1, 2, 4\}} := \{(0,1,9), (1, 9, 0), (3, 5, 7), (2, 4, 8)\}$ indicates that just these four 3-tuples are allowed concerning the three variables V_1, V_2, V_4. (Assuming that also the domains of the variables (the values -- 1-tuples) are given, and may be also some other constraints (or arbitrary arity (≤ 4)), the problem would be to compute $C_{\{1, 2, 3, 4\}}$, the set of all admissible 4-tuples obeying all the prevalent constraints.)

Roughly speaking, the general algorithm which serves as the basis of the reasoning technique reported here works as follows: Among the constraints given in an individual CSP it takes the ones with the highest arity, i.e., it first considers those restrictions which concern the largest number of variables. Then it propagates the relevant tuples to those lower-ary constraints which are influenced directly by these

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

specific (higher-ary) constraints. The interrelations are exemplified in fig 172. The constraint $C_{\{1, 2, 4\}}$ directly influences the constraints $C_{\{1, 2\}}$, $C_{\{1, 4\}}$ and $C_{\{2, 4\}}$. (The 3-tuples (of $C_{\{1, 2, 4\}}$) would get projected (in three different ways - depending on the (three) target constraint sets) according to 2-tuples belonging to the appropriate variables.) After having reached the bottom of the lattice an upward phase has to follow; ascending towards the top node finally computes the globally consistent solution ("n-consistency") after the arrival at $C_{\{1, n\}}$.

5.5.2 Alterations to the original algorithm

The algorithm described in the previous section has to be modified to obtain the data for future updates.

The descending phase "pre-filters" (deletes) the tuples that are not supplied by higher-ary constraints. For a following update it is necessary to know which tuples have been removed during the pre-filtering and which higher-ary constraints have been responsible for this deletion. For this goal, the updating algorithm introduces the so-called ΔC^d -sets. These sets consist of the tuples that have been pre-filtered and the constraints that have been responsible for their deletion. For unconstrained "constraints" that have been pre-filtered by a higher-ary - not unconstrained - constraint the ΔC^d -set will be marked (by •) as a set which consists of all tuples that are possible w.r.t. the given domains and not in one of the other sets (C-set or ΔC-set) - see below. Otherwise, if the higher-ary constraints are all unconstrained, the ΔC^d -set will be empty. In the case of real pre-filtering the tuples removed will be stored in the format $\Delta C^d = \{[t, \{C_1, \ldots, C_j\}], \ldots\}$ where t is a tuple not supported by the constraints $C_1, \ldots, C_j$ mentioned subsequently and $C_1, \ldots C_j$ are the higher-ary constraints (connected to the current one) responsible for the deletion of t. It is obvious that the highest constraint has an empty ΔC^d -set. Let us define the remaining tuples (the ones that have not been deleted) of each constraint to form the ΔC^d -sets. An exception are the one-ary constraints because they represent the domains of the CSP. Let us define the tuples obeying these constraints to be in the C-set. The next step is to perform adequate equi-joins (as in the database area) in an ascending order to get higher-ary tuples.

$$C^i = (C_1^{i-1} \bowtie \ldots \bowtie C_i^{i-1}) \cap \Delta C^i$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$\Delta C_{new}^{i} = \Delta C^{i} \setminus (C_1^{i-1} \quad \ldots \quad C_i^{i-1})$

The superscripts I and I-1 just mentioned denote the arity of a constraint. (Please note that every I-ary constraint at the stage I (>0) exactly has I downward edges to its related (I-1) ary constraints.)

The ΔC_{new} -sets will replace the old ΔC-sets after this operation since the old ones are no longer valid. The new ΔC-sets represent all the tuples that could not be derived during upward propagation. The C-sets maintain the consistent tuples for the sub-lattice headed by the computed constraint.

The updating algorithm works tuple-oriented, i.e., the algorithm performs an update only for changed tuples and for tuples affected by changed ones. The subsequent section introduces the procedure in greater detail; the example below already tries to sketch the main idea.

Consider the following (globally inconsistent) CSP concerning three variables:

$C_1 = C_2 = C_3 = \{0, 1\}$

$C_{1,2} = C_{1,3} = C_{2,3} = \{(0,1), (1, 0)\}$.

The descending phase has no effect; all ΔC^{d} -sets are empty. Thereby the ΔC-sets of the two-ary constraints are empty, i.e., all tuples of these constraints can be derived from the one-ary constraints. The ΔC-set of the highest constraint is marked as a $\bullet$-set (because it is originally unconstrained), and the constraint has an empty C-set since the equi-join of the two-ary constraints is empty.

Now let us restrict the constraint $C_{1,2}$ by removing the tuple (1, 0). The *pre-filtering update* procedure (in the next section) would produce the following:

$\Delta C_1^{d} = \{[1, \{C_{1,2}\}]\}$, $\Delta C_2^{d} = \{[0, \{C_{1,2}\}]\}$.

Subsequently, the ascending update produces:

$\Delta C_{1,3} = \{(1, 0)\}$, $C_{1,3} = \{(0, 1)\}$

$\Delta C_{2,3} = \{(0, 1)\}$, $C_{2,3} = \{(1, 0)\}$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The highest-ary constraint $C_{1,2,3}$ remains unchanged since a further restriction cannot alter global inconsistency.

Suppose we now add the tuple (1, 1) to the constraint $C_{2,3}$: this relaxation has the following effects:

C_2, ΔC_2^d, C_3, ΔC_3^d will not be affected

$\Delta C_{2,3} = \{(0, 1)\}$ $C_{2,3} = \{(1, 0), (1, 1)\}$

$\Delta C_{1,2,3}$: • $C_{1,2,3} = \{(0, 1, 1)\}$

(So, in this case, global consistency could be achieved - $C_{1,2,3} \neq \{\}$.)

This example should afford a short insight into the working method of this algorithm. In the following, a better view is presented.

5.5.3 The updating Algorithm

The algorithm supports two different types of updating operations:

- **restriction:** a formerly valid tuple will get removed from the constraint.
- **relaxation:** a formerly unsupported tuple will be added to a constraint.

All possible varieties of updating can be modeled using these two operations.

5.5.3.1 Restriction

Let t be the tuple to be removed from the constraint C. The restriction needs the following operation on C to be done:

1. **If** $t \in \Delta C^d$ **then** delete t from ΔC^d, stop.

2. **If** $t \in \Delta C$ **then** delete t from ΔC, continue with pre-filtering update - as below.

3. **If** $t \in C$ **then** delete t from C, continue with pre-filtering update.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The pre-filtering update process works as follows:

(Let t be the tuple derived by projecting t onto the components indicated by the index-set of the specific constraint.)

For all constraints downwards connected to the updated constraint **and** for which the following (*) holds **do**:

(*) $C_{temp}^{(i)} := C^{(i)} \cup \Delta C^{(i)}$,

$C_{temp}^{(i-1)}$ is $C_{temp}^{(i)}$ projected according to the index-set of a constraint connected downwards.

If $\bar{t} \notin C_{temp}^{(i-1)}$ **then do** the following:

1. **If** $\bar{t} \in \Delta C^d$ **then** stop pre-filtering update for this sub-lattice. (There is no support for this tuple in the sub-lattice underneath.)
2. **If** $\bar{t} \in \Delta C$ **then** delete $\bar{t}$ from ΔC and add it to ΔC^d in the format described in the previous section preserving the constraint not supporting the tuple, pre-filtering update has to be invoked w.r.t. this sub-lattice.
3. **If** $\bar{t} \in C$ **then** delete $\bar{t}$ from C and add it to ΔC^d (see the recent description), store both the constraint and the removed tuples in a temporary list L, pre-filtering update has to be invoked w.r.t. this sub-lattice.

Until no more unprocessed constraints in the sub-lattice denoted by the updated constraint exist or the bottom of the CSP structure has been reached.

The list L generated during the pre-filtering update will be necessary for the process starting now:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

For each constraint stored in the list L **do:**

(The constraints stored in L are being processed in ascending order w.r.t. their arities.)

1. **For each** of the constraints upwards connected to the stored constraint **do**:
   (a) **For each** removed tuple t stored with the constraint **do**:

      i. Delete all occurrences of $\hat{}_t$ in the C-set and add the deleted tuples to the ΔC-set of the constraint currently processed. ($\hat{}_t$ is the tuple t expanded according to the index-set of a higher-ary constraint.)
      ii. **If** the previous operation has changed the C-set **then** include the processed constraint and the deleted tuples in the list L w.r.t. the constraint's arity.

   Until the list L has completely been processed.

   After this procedure, we get global (in-)consistency.

5.5.3.2 Relaxation

The relaxation is being handled nearly the same way the restriction is.

Let t be the tuple being added to the constraint C.

Let $\bar{t}$ be the tuple t after projection according to the index-set of the actual constraint.

Define the temporary list L as being empty.

For each constraint downwards connected to C **do:**

1. **If** $\bar{t} \in \Delta C^d$ **then**

   begin

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Delete the constraint connected upwards from the set associated with this tuple in the ΔC^d -set.

If this set is now empty **then** delete the tuple from the ΔC^d -set and add it to the ΔC-set, include the

constraint and the new tuple added to the ΔC-set at the end of the list L, pre-filtering update has to

be invoked w.r.t. this sub-lattice.

End

2. **If** $\bar{t} \in \Delta C$ **then** stop pre-filtering update for this sub-lattice.
3. **If** $\bar{t} \in C$ **then** stop pre-filtering update for this sub-lattice.

Until all constraints in the sub-lattice denoted by the updated constraint have been processed or the bottom of the CSP structure has been reached.

After having applied this task to the CSP we have gained the list L with all constraints updated via pre-filtering and their new tuples. We will achieve global (in-)consistency by applying the following steps to this list:

For all constraints stored in the list L **do:**

1. **For all** the new tuples $\bar{t}$ in the ΔC^i -set check whether

$$\bar{t} \in (C_1^{i-1} \bowtie \cdots \bowtie C_i^{i-1})$$

If this case happens to be true delete the tuple $\bar{t}$ from the ΔC-set, add it to the C-set, and include the constraints connected upwards to the updated constraint in the list L w.r.t. their arity.

2. **For all** new tuples t in $(C_1^{(i-1)} \bowtie \ldots \bowtie C_i^{(i-1)})$ check whether

$$t \in \Delta C^{(i)}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Should this be the case the same operations must be done as described in the previous item.

Until every entry in the list L has been processed.

This achieves global (in-)consistency.

5.5.4 Examples

(Parentheses around 1-tuples are omitted.)

Consider the following CSP w.r.t. (n:=) 4 variables:

$C_1 = C_2 = C_3 = C_4 = \{0, \ldots, 9\}$

$C_{1,2} = \{(0, 1), (3, 7), (6, 8)\}$

$C_{3,4} = \{(0, 0), (2, 2), (5, 9)\}$

$C_{1,2,3} = \{(0, 1, 2), (2, 4, 6), (6, 8, 0)\}$

The following sets represent the result produced by the algorithm described before along with the alterations presented before.

Please recall that • denotes the case that the set maintains all the tuples which are possible concerning the given domains not belonging to one of the other sets.

$\Delta C_{1,2,3,4}^{d} = \varnothing$

$\Delta C_{1,2,3,4} = \bullet$

$C_{1,2,3,4} = \{(0, 1, 2, 2), (6, 8, 0, 0)\}$
(Here we find the global solution.)

$\Delta C_{1,2,3,4}^{d} = \varnothing$

$\Delta C_{1,2,3} = \{(2, 4, 6)\}$

$C_{1,2,3} = \{(0, 1, 2), (6, 8, 0)\}$

$\Delta C_{1,2,4}^{d} = \varnothing$

$\Delta C_{1,2,4} = \bullet$

$C_{1,2,4} = \{(0, 1, 0), (0, 1, 2), (0, 1, 9), (6, 8, 0), (6, 8, 2), (6, 8, 9)\}$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$\Delta C_{1,3,4}{}^{d} = \varnothing$

$\Delta C_{1,3,4} = \bullet$

$C_{1,3,4} = \{(0, 2, 2), (6, 0, 0)\}$

$\Delta C_{2,3,4}{}^{d} = \varnothing$

$\Delta C_{2,3,4} = \bullet$

$C_{2,3,4} = \{(1, 2, 2), (8, 0, 0)\}$

$\Delta C_{1,2}{}^{d} = \{[(3, 7), C_{1,2,3}]\}$

$\Delta C_{1,2} = \varnothing$

$C_{1,2} = \{(0, 1), (6, 8)\}$

$\Delta C_{1,3}{}^{d} = \bullet$

$\Delta C_{1,3} = \{(2, 6)\}$

$C_{1,3} = \{(0, 2), (6, 0)\}$

$\Delta C_{1,4}{}^{d} = \varnothing$

$\Delta C_{1,4} = \bullet$

$C_{1,4} = \{(0, 0), (0, 2), (0, 9), (6, 0), (6, 2), (6, 9)\}$

$\Delta C_{2,3}{}^{d} = \bullet$

$\Delta C_{2,3} = \{(4, 6)\}$

$C_{2,3} = \{(1,2), (8, 0)\}$

$\Delta C_{2,4}{}^{d} = \varnothing$
$\Delta C_{2,4} = \bullet$

$C_{2,4} = \{(1, 0), (1, 2), (1, 9), (8, 0), (8, 2), (8, 9)\}$

$\Delta C_{3,4}{}^{d} = \varnothing$

$\Delta C_{3,4} = \{(5, 9)\}$

$C_{3,4} = \{(0, 0), (2, 2)\}$

$\Delta C_{1}{}^{d} = \{[1, \{C_{1,2}, C_{1,3}\}], [2, \{C_{1,2}\}], [3, \{C_{1,2}, C_{1,3}\}], [4, \{C_{1,2}, C_{1,3}\}],$
$[5, \{C_{1,2}, C_{1,3}\}], [7, \{C_{1,2}, C_{1,3}\}],$
$[8, \{C_{1,2}, C_{1,3}\}], [9, \{C_{1,2}, C_{1,3}\}]\}$

$C_{1} = \{0, 6\}$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$\Delta C_2^{\,d} = \{[2, \{C_{1,2}, C_{2,3}\}], [3, \{C_{1,2}, C_{2,3}\}],$
$[5, \{C_{1,2}, C_{2,3}\}], [6, \{C_{1,2}, C_{2,3}\}],$
$[7, \{C_{1,2}, C_{2,3}\}], [9, \{C_{1,2}, C_{2,3}\}]\}$

$C_2 = \{1, 8\}$

$\Delta C_3^{\,d} = \{[1, \{C_{1,3}, C_{2,3}, C_{3,4}\}],$
$[3, \{C_{1,3}, C_{2,3}, C_{3,4}\}],$
$[4, \{C_{1,3}, C_{2,3}, C_{3,4}\}],$
$[5, \{C_{1,3}, C_{2,3}\}],$
$[6, \{C_{3,4}\}],$
$[7, \{C_{1,3}, C_{2,3}, C_{3,4}\}],$
$[8, \{C_{1,3}, C_{2,3}, C_{3,4}\}],$
$[9, \{C_{1,3}, C_{2,3}, C_{3,4}\}]\}$

$C_3 = \{0, 2\}$

$\Delta C_4^{\,d} = \{[1, \{C_{3,4}\}], [3, \{C_{3,4}\}],$
$[4, \{C_{3,4}\}], [5, \{C_{3,4}\}],$
$[6, \{C_{3,4}\}], [7, \{C_{3,4}\}],$
$[8, \{C_{3,4}\}]\}$

$C_4 = \{0, 2, 9\}$

Now the C-sets maintain the i-consistent tuples; so, $C_{1,\ldots,4}$ contains the globally consistent solution ("n-consistency").

The following part provides an updating example:

Let $C_{1,2,3}^{new} := C_{1,2,3} \cup \{0, 1, 5)\}$; so (0, 1, 5) is a new tuple added to the constraint, i.e., the constraint is being relaxed.

The update of the pre-filtering sets C^d looks as follows:

The new tuple is not being filtered by a constraint connected upwards; therefore, we place it in the $\Delta C_{1,2,3}$ set.

$\Delta C_{1,2,3}^{new} = \{(2, 4, 6), (0, 1, 5)\}$

The next step is to update all the constraints connected downwards ($C_{1,2}$, $C_{1,3}$, $C_{2,3}$):

$(0, 1) \notin \Delta C_{1,2}^{\,d} \cup \Delta C_{1,2} \cup C_{1,2} \Rightarrow$ no change for $C_{1,2}$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$(0, 5) \in \Delta C_{1,3}^{d} \Rightarrow \Delta C_{1,3}^{new} = \{(2, 6), (0, 5)\}$

$(1, 5) \in \Delta C_{2,3}^{d} \Rightarrow \Delta C_{2,3}^{new} = \{(4, 6), (1, 5)\}$

Now we do the update for the one-ary constraints:

$0 \in C_1, \quad 1 \in C_2 \Rightarrow$ no change

$5 \in \Delta C_3^{d} \Rightarrow$ delete $[5, \{C_{1,3}, C_{2,3}\}]$

$C_3^{new} = \{0, 2, 5\}$

The ascending phase produces the following changes:

$C_{1,3}^{temp} = \{(0, 5), (6, 5)\} \Rightarrow$
$C_{1,3}^{new} = \{(0, 2), (6, 0), (0, 5)\}$
$\Delta C_{1,3}^{new} = \{(2, 6)\}$

$C_{2,3}^{temp} = \{(1, 5), (8, 5)\} \Rightarrow$
$C_{2,3}^{new} = \{(1, 2), (8, 0), (1, 5)\}$
$\Delta C_{2,3}^{new} = \{(4, 6)\}$

$C_{3,4}^{temp} = \{(5,0), (5, 2), (5, 9)\} \Rightarrow$
$C_{3,4}^{new} = \{(0, 0), (2, 2)\}$

$C_{1,2,3}^{temp} = \{(0, 1, 5)\} \Rightarrow$
$C_{1,2,3}^{new} = \{(0, 1, 2), (6, 8, 0), (0, 1, 5)\}$
$\Delta C_{1,2,3}^{new} = \{(2, 4, 6)\}$

$C_{1,3,4}^{temp} = \{(0, 5, 9)\} \Rightarrow$
$C_{1,3,4}^{new} = \{(0, 2, 2), (6, 0, 0), (0, 5, 9)\}$

$C_{2,3,4}^{temp} = \{(1, 5, 9)\} \Rightarrow$
$C_{2,3,4}^{new} = \{(1, 2, 2), (8, 0, 0), (1, 5, 9)\}$

$C_{1,2,3,4}^{temp} = \{(0, 1, 5, 9)\} \Rightarrow$
$C_{1,2,3,4}^{new} = \{(0, 1, 2, 2), (6, 8, 0, 0), (0, 1, 5, 9)\}$

So the globally consistent solution could really get enlarged.

This updating procedure discussed above contains two important features: it allows soft modifications (because it works tuple-oriented - instead of adding or deleting entire constraints), and it locally performs just the necessary changes to compute the global solution as quickly as possible. It thereby allows cooperative work both among computers and between humans and computers.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The necessity of a thought-out mechanism for the appropriate handling of changes is obvious, and it has a broad range of application; for instance, it is particularly welcome in the deductive database area.

5.6 Artificial Intelligence for cooperative Information systems

5.6.1 Cooperative Information systems

Within the natural-language research community it has long been acknowledged that the conventions and pragmatics of natural-language communication often oblige dialogue systems to consider and address in their responses the underlying purposes of queries rather than replying to them literally and without further comment or elaboration. These systems cannot simply translate queries into transactions on database or expert systems, but must apply many more complex reasoning mechanisms to the task of selecting appropriate and useful responses. Researchers argued that if dialogue systems are to be perceived by their users as acceptable partners in conversation, they must be able to provide extended, cooperative responses that may include many general elements as:

- the so-called direct answer: the information literally requested.
- information or action related to the direct answer.
- information or action in place of the direct answer;
- information or action that is related to one or more of the questioner's stated, inferred, or assumed goals.

Guided by these sorts of intuitions, people have started investigating cooperative dialog from a wide variety of Artificial Intelligence perspectives, including discourse representation and processing, plan recognition, rational agency, natural-language generation and cooperative agents. Of course, cooperative agents & cooperative dialog is a very new field and a lot of work is stored for the future than what has been done. Although our topic is not natural-language analysis, but cooperative responses have plentiful applications in database-query applications, distributed task-domain activities, domain-independent principles of conversation.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Database-query applications

The domain of database queries has long been a popular application area for natural-language processing technology. This is largely because (1) the query task is well understood and in many cases is supported by a well-developed mathematical theory; (2) the database structure provides a complete and precise definition of the domain of discourse; and (3) the majority of the queries can be handled with relatively small lexicons and grammars.

Users of database-query systems are modeled as simple information seekers, engaging the system in a dialog solely for the purpose of obtaining information. Any underlying purpose they might have in obtaining information is beyond the scope of the system's reasoning, influencing neither the interpretation of the question nor the presentation of the answer. The user provides the system with a natural-language characterization of the desired result set - the elements to be retrieved from the database and also specifies the requirements for how that result set is to be presented. The system then analyzes the user's input and dynamically constructs a procedure that, when executed in conjunction with the database, is expected to retrieve all the elements fitting user's characterization. This set is the *direct answer* to the query.

Research had focused on the problem of mapping natural-language structures onto formal query languages and on related issues such as transportability of interfaces across databases. More recently, it has been noticed that the connection between a query and appropriate answer is not a direct one, and hence database interfaces should offer extended answers.

Kaplan showed need of such elaborate answers [89] [90] in the COOP system where he designed to detect and correct *false extensional presumptions* on queries. Others later came up with algorithms for correcting *intensional* presumptions, *query generalization* to detect and correct misconceptions in queries and *integrity constraints* to produce more helpful and informative answers. All these efforts were to modify or augment the direct answer such that it becomes more acceptable to the user.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Task-domain applications

Database-query applications have no explicit knowledge of the user's *goal* and thus cannot choose answers based on considerations of *why* a given query is being asked. But the true nature of cooperation seems to relate at least partially to how agents help each other achieve goals. This implies that database-query applications are able to behave "cooperatively" only to the extent that their designers are able to build into the systems their assumptions about the kinds of goals users typically have in the world and which motivate their use of the systems.

In *task-domain applications*, the execution of some underlying task is central to the purpose of the dialog. Two common task-domain applications are *intelligent help* systems and *expert advisors*. The first one is used to help people solve problems in some domain (e.g. to send am e-mail to another user), the second one to help people make decisions (e.g. to assist in the development of an investment strategy.) These types of applications must explicitly represent their goals and reason about their progress towards in order to help users complete their tasks in the domain.

Again, significant work has been done in detecting and correcting misconceptions in the intensional or extensional pre-assumptions. Task-oriented domains are highly more complex than database-query domains, and researchers in this area have been addressing correspondingly complex questions, including:

- For a given domain and application, what sorts of response behavior are perceived as cooperative?
- What principles can be used to organize information in cooperative responses?
- What forms of knowledge representation and what reasoning methods are necessary to support the production of cooperative behavior?
- How do dialogue systems acquire the knowledge needed to exhibit cooperative behavior?

Principles of conversation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Grice in his paper *logic and conversation* [92] defined *Cooperative Principle* (CP) and its four related maxims as *quality, quantity, relation and manner* which spawned many investigations into cooperative processes. He tried to reconcile formal approaches and informal approaches to linguistic theories of meaning in his paper and he mentioned that the CP and the four maxims in broad descriptive terms are not directly suitable for computer implementation.

The AI community has been searching for computational counterparts to the principles of Gries in context of specific domains or applications. Some researchers have proposed domain-independent principles of cooperative dialog. One important work is Joshi's work on the principle of "squaring away mutual beliefs[93]" and Joshi et al on preventing false inferences[94], living upto users' expectations [95] and anticipating users' default inferences [96, 97]. For example, plan recognition is a very general reasoning process that invokes cooperative behavior in conversation. Allen developed the first formal model of plan recognition [98] and other researcher extended it to handle multiple input turns. Others re-analyzed plan recognition in a belief-based framework to permit inference of ill-formed plans. Recent efforts have moved to understand plan recognition within the larger framework of discourse interpretation. Since Grosz and Sidner's foundational work on the structure of discourse [99], theorists have been designing formalisms for modeling the beliefs, intentions, inferences and change over the course of a conversation. They have borrowed ideas from plan recognition in an expanded theory of discourse understanding based on a view of conversation as collaboration.

Recently, Galliers proposed a formal model of rational agenthood and cooperative interaction based on a version of speech-act theory. Two noteworthy characteristics of her model are: (1) it is based on a logic of belief and intention, which brings it into the sphere of computable accounts of cooperation, and (2) it acknowledges the significance and value of *conflict* in cooperation among agents, and thus makes no assumptions regarding agents' sincerity or benevolence. The theory, however, has not been applied in the building of a functional dialogue system, and thus its prescriptive power is unknown.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.6.1.1 Progress towards a computational theory of cooperative dialogue

There are two goals one may imagine for research in cooperative dialogue: (1) to develop technology with which to build "friendly", intuitive and easy-to-use information systems, and (2) to discover the underlying principles, representations and reasoning procedures that give rise to the kinds of conversational behavior that people find acceptable and "co-operative". Research has been more successful in advancing the first goal, but with the caveat that few methods show potential to robustly transcend the example scenarios for which they were originally designed. Progress towards the second goal, however has been marginal. While the discourse representations currently under development show some promise, they stop well short of the point at which cooperative reasoning begins.

5.6.1.1.1 Theories derived from examples

A common approach in the study of cooperative dialogue can be called the *example method*. Here the researcher starts by characterizing a domain of discourse and a hypothetical AI application within that domain, such as a natural-language interface to a database or an expert advisor. Next, an example is selected and used as a method of circumscribing a class of response phenomena. Examples typically consist of a reasonable query in the domain followed by a pair of alternative responses, one of them designated "uncooperative" and the other "cooperative" and so preferred. The research problem has two parts: first, a general explanation for the preference must be given, and second, an algorithm must be designed which will enable the hypothetical AI application to choose the "cooperative" solution as opposed to the unpreferred one.

But such use of example method has produced systems deficient in the following important respect:

they lack explicit representations of the goals they are attempting to achieve by means of their responses. In effect, the systems do not model the reasoning and decision-making processes through which cooperative discourse partners choose their responses, but rather model *situations* in which selected forms of response behavior are *asserted* to be "cooperative".

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Kaplan's COOP as case study: One of the better illustrations of the example method and its consequences is the COOP system designed by Kaplan [91]. Working in the domain of natural-language database queries, Kaplan identified the problem of *extensional presumption failure*, demonstrated in the example (1) below.

(1) Q: Which programmers from the Windows group are in department 67?

Ans:

A. None.

B. I don't know of any programmers in department 67.

Kaplan argues that the query falsely presumes that the set of "programmers in department 67" is non-empty, where he defined a *presumption* as a proposition whose falsehood entails that at most one correct direct answer exists. (A *presupposition*, in contrast, is a proposition that must be true for any direct answer to be meaningful.) Kaplan suggested that when people pose questions, they normally do so with the belief that they have left the respondent with a choice of direct answers. Since a false resumption implies no such choice, it can be taken as evidence that the questioner holds an incorrect belief. Thus the answer in A would be misleading because it implicitly confirms the incorrect belief. Response B is preferred because it directly addresses the questioner's misconception.

Kaplan recognized that false presumptions of database queries could be discovered by testing the constituent sets for emptiness. Note that the direct answer to the query in (1) is derived from the following five sets:

1. The set P of all programmers.
2. The set W of all members of the Windows group.
3. The set D of all members of department 67.
4. The set PW of programmers in the Windows group;

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5. The set PD of programmers in department 67.

The direct answer is obtained from the intersection of PW and PD. If PD is empty in the current world as represented by the database, the empty set results. When COOP finds that the answer to the query is an empty set, it checks the constituent sets for emptiness. Any empty sets found are described to the user in place of the direct and misleading response "None". More precisely, COOP presented only the *least failing sets*, those sets from which all other empty sets can be derived. It is important to note that COOP neither represents nor reasons about the goals in the conversation. Its behavior can be best described as *stimulus-response*, that is, when stimulated by a null-query result, it responds by testing subsets of the query for emptiness. Its theory of cooperation consists of the simple dictum "correct the user's false presumptions".

The important point is that COOP should not be viewed as a model of cooperative response behavior, but rather as a model of a class of situations in which one particular type of response is more intuitive that the other.

5.6.1.1.2 Theories of representations and reasoning

Although most research on cooperative dialogue uses the example methods, all of it is not limited that way. In this category, studies of discourse representation and processing, including plan recognition, deserve consideration. Existing theories of discourse stop short of describing how cooperative dialogue participants use their discourse models when deciding how to reply. The theories explain the nature of the representations and inference processes that underlie cooperative discourse, but do not actually go into the nature of cooperative behavior itself. To understand this issue, consider the dialogue shown in (2), which has recently appeared as a paradigm example on discourse modeling.

(2) A: I want to lift the piano.

B: OK.

A: On the count of three, I'll pick up this [deictic to foot] end, and you pick up that [deictic to board] end.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

B: OK.

A: One, Two, Three!

Here two agents collaborate in order to achieve a shared goal of lifting a piano. Two questions of interest working on models of discourse are those:

1. What beliefs and intentions must the respective dialogue participants hold in order for them to collaborate successfully?
2. How do the participants' utterances affect their respective discourse models?

What this work does not address is the connection between a participant's discourse model and her choice of response. That is, while discourse theories such as the one advanced by Grosz and Sidner [99, 100] strive to model the effects of A's first utterance on B's beliefs and intentions, they do not explain how B's modified beliefs and intentions lead her to respond with an "OK" without following up with more statements. The point is that not only one should explain why B chose to accept the shared plan, but one must explain how that choice affects her decisions of *what to say* in her response and *how to say it.*

5.6.1.1.3 the "theory gap"

So, we notice two issues: on one hand, efforts to model examples of cooperative response behavior have created procedures for computing helpful and informative responses, the discourse settings to which the procedures are applicable lack generality. On the other hand, theories of discourse processing have advanced our understanding of the beliefs and intentions that participants in conversation adopt and act upon, but they have little to say how the cooperative responses are computed based on those beliefs and intentions.

There is a "theory gap" that exists between accounts of the beliefs and intentions that discourse participants acquire and modify during the course of a conversation and accounts of the responses they generate.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.6.1.2 The respondent-based perspective as a framework for progress

Cooperative response generation will probably tend in two directions: (1) developing context-sensitive techniques to improve system response behavior, and (2) formalizing the representations and inference methods that seem to support cooperative dialogue. The results of these efforts are of limited usefulness to designers of next generation ICISs because they typically account for examples of cooperative behavior, adding little to our understanding of the general nature of cooperation in dialogue.

Principles of cooperative behavior can be defined from two perspectives, the *questioner-based* perspective has been the dominant paradigm in the past. Now a shift in perspective from questioner-based perspective to respondent-based perspective is necessary for achieving further progress in understanding cooperative dialogue.

5.6.1.2.1 Questioner and respondent-based perspectives

A model of cooperative response generation is *questioner-based* if it characterizes cooperative interaction in terms of how the system's responses affect the questioner in a particular query. All judgments of cooperation today are made from the questioner's point of view, i.e. a response is "cooperative" in a given situation if and only if it is perceived as such by the questioner. A fundamental premise of questioner-based investigations is that a questioner in a given dialogue situation has various needs, desires, preferences and expectations, some of which may be directly evident in his query and its accompanying utterances, and others of which may require more complex reasoning chains to detect. A questioner may judge a dialogue system *uncooperative* if it fails to meet his needs, satisfy his desires, accommodate his preferences and live up to his expectations in a particular query situation.

Previously-developed models of cooperative response behavior have been questioner-based. Their behavioral goals with respect to the questioner have included avoiding adverse effects on the questioner's beliefs about the world (e.g. preventing the questioner from drawing incorrect conclusions from the system's response), improving an interface's ease of use (e.g., by anticipating follow-up questions so that

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the user need not ask them explicitly), and adding useful functionality (e.g. offering to provide information in the future when it becomes available). In all these cases, the aim of the work was to design modifications of the system's default response behavior in order to better meet the needs of its users.

In contrast, a model of cooperative response generation is *respondent-based* if it characterizes both the knowledge and the decision-making processes *of the respondent* that together account for her choice of cooperative response in a given query situation. Respondent-based models attempt to define the respondent's role as a cooperative conversational partner.

Respondent-based models take seriously the view of language as action that forms the foundation of speech-act theory. To do so, they adhere to the principle that responses reflect the respondent's goals. It follows from this principle that respondent-based cooperative response generation systems must be designed as *planners*, systems that are able to *identify, adopt, plan to achieve and possibly modify* their response goals. Here are three example problems that can only be solved in a respondent-based theory.

5.6.1.2.2 Three challenges for ICIS design

This section discusses three challenges for ICIS design: (1) a respondent's ability to be cooperative is a function of her knowledge, rather than an objective notion; (2) co-operativeness is related to the amount of effort a respondent expends on the questioner's behalf; and (3) respondents need to reason about appropriate limits on the information content of their responses.

Cooperativeness is knowledge-limited: In questioner-based accounts, "cooperative" is treated as an absolute notion, that is, a response is either cooperative or uncooperative. The criteria used for determining whether a given response is cooperative or not are stated from the questioner's point of view.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The problem with this formulation is that a respondent's ability to respond cooperatively varies with the quantity, quality and detail of her knowledge. Compare the two responses in example (3) below, taken from the transcripts of e-mail messages sent between users and staff consultants.

(3) Q: How do you change mode for a dotted file (such as .login)?

A: You would use the same method as for any other file you own. Basically, just type "chmod mode filenames" [....]

B: The problem may be that you don't own your .login file. To save space, most of the .login files are linked together (i.e. they are all one file). To fix this, type the following: [...]

In this example, two *different* consultants respond to the same query. Respondent A evidently concluded that Q had a misconception that different techniques are used to change the mode of "non-dotted" and "dotted" files. Respondent B, however, possessed knowledge that A seems to have lacked the knowledge that , .login files are linked together and thus not owned by Q. This knowledge profoundly affects her choice of response. Note, however, that we would not like to say that A's response is uncooperative. Rather she behaved as cooperatively as she could *within the limits of her knowledge*.

A questioner-based approach could at-best provide independent accounts of A's and B's responses. That is, each response would be treated as a kind of pattern, representative of a class of similar responses. Then the research would proceed to identify the knowledge and reasoning mechanisms needed to generate instances of each response class. What we desire, however, is a single theory explaining both responses. By viewing the responses as actions planned to achieve goals, it should be possible to show exactly how differences in the two respondents' knowledge lead them to select different, yet equally cooperative responses from their points of views.

Cooperativeness and amount of effort:

Another intuition that questioner-based accounts fail to account for is that judgments of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

cooperativeness seem to correlate with questioners' perceptions of the *amount of effort* that has been expended on their behalf. One possible explanation, then, for why current systems appear cooperative is that their responses have unwittingly been constructed in a way that leads to the perception that the systems have expended as *much effort as could be reasonably expected of them to* fully address the query within their limited domains of reasoning. Consider the following naturally-occurring example:

(4) Q: Is it possible to log into cory through the port selector? I Need to use the graphics terminals

at Evans.

A: No. Cory is not on the CF&O port selector.

What is it that makes the response seem less than fully cooperative? One explanation is that it fails to explicitly address the questioner's stated goal of using the Evans graphics terminals, and thus gives the impression that R did not make an adequate effort to help Q solve his problem.

While improving the response behavior in the example above may be a relatively straightforward matter, the problem for future ICIS designers is that in complex domains, it simply may not be possible to address all the explicit and implicit needs of the query. To cope with this, systems will need to possess knowledge of their limitations and be able to reason with that knowledge to decide when to offer apologies, explanations for inadequate responses, and so forth. This sort of reasoning is essentially respondent-based, since the respondent must reason about her own limitations and the questioner's perception of her response in order to satisfy her own goal of maintaining the appearance of cooperativeness. That is, we might hypothesize a basic respondent-based cooperative principle of the form "Respond in a manner that supports the questioner's perception that reasonable effort has been expended", where the setting for reasonable effort is context-dependent, affected by the interpersonal relations between the questioner and the respondent.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The problem of quantity: The maxim of quantity can be formulated as follows: "Make your contribution as informative as is required; do not make your contribution more informative than is required." This maxim reflects the intuition that in any given response-generation situation, there exists both lower and upper bounds on the information content of the response.

Systems in the past have not had to reason about how much had to be said in response or how much was too much. Confining the task to modeling particular examples made it possible to use the examples as patterns which reflected appropriate choices of upper and lower bounds. But as designers of ICISs attempt to tackle more complex applications such as intelligent help and expert advisor systems, they will have to develop explicit principles governing the choice of lower and upper bounds for information content of responses. Such principles can be naturally incorporated in a plan-based framework and used to guide the system's choice of response goals.

5.6.2 Knowledge architecture in Cooperative Information systems

As complex systems cover heterogeneous domains whose knowledge and associated representations are controlled by complex activities. To handle sound complexity an open framework is crucial. Proper choice of sophisticated formalisms and tools which will capture the knowledge of the domains and use them , definition of cooperative activities to determine the complex goal etc. will make a great difference in the power of the cooperative information system. In general, knowledge is distributed over knowledge-bases (KBs) which are grouped around societies (SKBs). A single knowledge base is specialized in a specific problem solving. It consists in local production rules (Ri) and local facts basis (Fi). The knowledge-base also encapsulates the specific resolution strategy. The strategy is determined on a distributed basis through negotiation method using a conversation through a separate channel of cooperation.

A society of KBs (SKB) collaborate to achieve a task. An instance of an SKBi may be seen as a multi-agent system. An SKB is an union of a few KBs which is a partition of the global knowledge-base. An SKB may be organized in two ways:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(1) Vertical hierarchy. Its elements (KBs) may share task according to the task-sharing protocol. In this case, a specific module, called a manager allocates a task to each module of the society which is called the worker. It manages activation, termination of operations, according to the well known client-server model, with the manager as the client and the worker as the server.

(2) Horizontal partnership. Modules (Kbi) may know each other and exchange information according to a result-sharing protocol.

An SKB communicates with the control module in two cases: to provide execution results, which are put up on the shared knowledge space, or to request a service from the control, for instance, when it fails. The communication is in units of events and messages. Events are reported to inform current states and transitions, messages are sent to request services. The overall knowledge architecture model appears in fig 173.

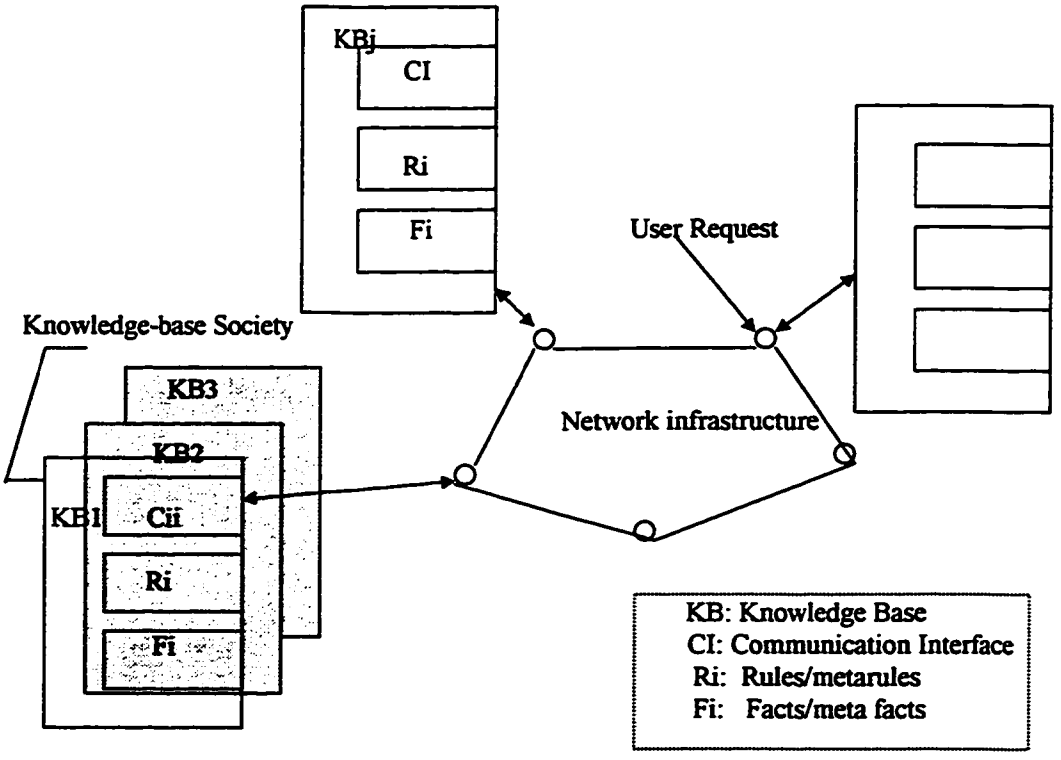


Figure 173:Modeling of Distributed Cooperative Information Systems

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

This architecture model describes an open information system in terms of the services it offers and those it requires from the environment. At highest abstraction (i.e. root), offered services formalize system functionality through, for which a user addresses requests. Upon receipt of request, the system activates adequate processing, based on domain knowledge, noted meta facts and the reasoning on that domains, expressed by means of meta rules. These informations are distributed over cooperative knowledge societies. The underlying model supports hierarchical descriptions by stepwise service specialization. It comes down to reduce service's complexity by defining sub-services handling more manageable situations, with major criteria to reduce inferences on a restricted knowledge-base defined by its role in the application. The knowledge-base at a node is again composed of some basic units called modeling-units which is to described now.

5.6.3 Modeling_Unit Structure

A modeling_unit is an abstraction of a real-world entity, which is essentially described by two components: a list of attributes and a list of services as shown in fig 174.

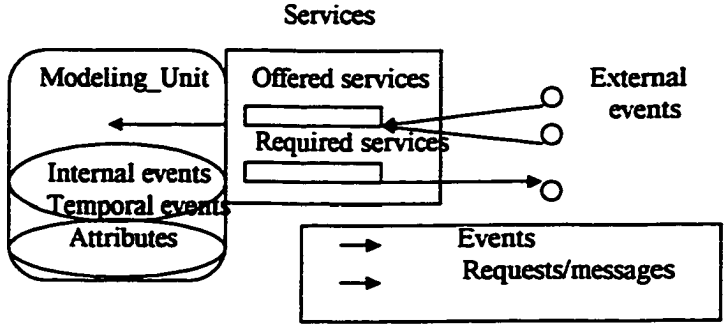


Figure 174:Modeling_Unit structure

An attribute is an abstraction of a single characteristic possessed by an entity that is itself abstracted as a Modeling_Unit. An attribute may be simple (name string, number, float) or complex when it represents reference to another Modeling_Unit identity. Broadly speaking, attributes are local facts manipulated by the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

entity. A service is provided by a Modeling_Unit to its environment. Services may be divided into provided and required. A detailed study of system components responsibilities allows to define provided and required services.

Services definition imply several communication features within a system. These features are commonly known as protocols and help to structure information flows causally related over time. For instance, it is interesting to know the necessity of broadcasting when requiring a service, to know if acknowledgments of services are required, and if these acknowledgments are blocking or non-blocking. These requirements are important as some activities may be in parallel. A Modeling_Unit has a type and an instance. The type encapsulates all unit properties definition i.e., attributes, services, relationships and events. At the instance level, a Modeling_Unit has a state which can change according to the occurrences of events. Besides the reference to a type, the formalism allows for sub typing. Sub typing describes a specialization of a Modeling_Unit. Moreover, a type may have many subtypes. This property allows to handle derivation, from a given formalism, of multiple variants sharing type properties, each of which is associated with its own properties and proof techniques. We want to retain the *part of* relationship to describe complex Modeling_Units. This concept allows to express structural links among a Modeling_Unit society whose behavior is strongly coupled. For instance, the management department of a bank is a sub-society of a bank, which has its own services and attributes. However its structure is part of the bank structure and its behavior is strongly coupled to that of the associated bank. Specifying this as a standalone Modeling_Unit is not judicious, as the amount of information flows between that Modeling_Unit and remaining parts of the world is negligible. It is more natural to consider it as a part of the bank Modeling_Unit.

Services may be accessed by Modeling_Units with which the considered unit is involved in a relationship, it may also be accessed by any other Modeling_Unit without underlying structural relationship. For instance, services of a Modeling_Unit Clock may be invoked freely from structural relationships. Let us retain the *part of* relationship to describe complex Modeling_Units. This concept allows to express structural links among a Modeling_Unit society whose behavior is strongly coupled.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.6.4 Modeling_Unit Behavior

System's behavior is specified in terms of events occurrences and subsequent service activation. A service execution implies elementary or complex transformations applied on some elements of the attribute component. Transformations imply eventual message exchanging to notify a treatment, or an acknowledgment, or a processing result.

Event

An event occurs when a noteworthy change happens in a Modeling_Unit. The processing to perform this change is embedded in the Modeling_Unit, together with its pre and post conditions. Three kinds of events are distinguished:

- External events stimulated by the environment.
- Internal events usually due to an object change.
- Temporal events which may occur at certain dates in the system.

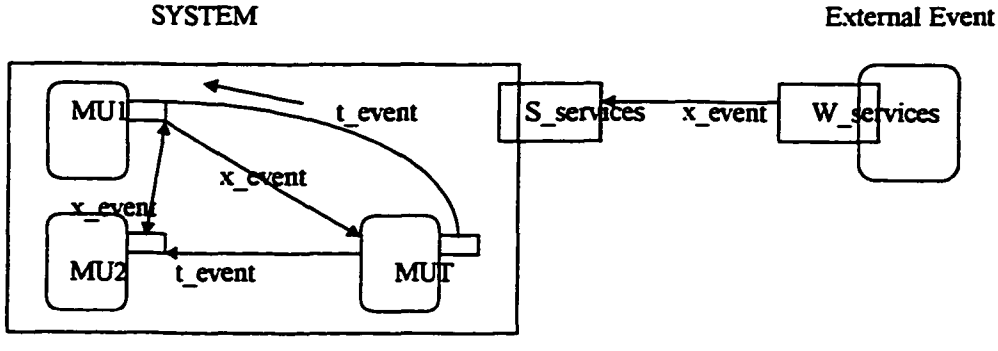


Figure 175:External and Temporal events

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

External events may be triggered by society's neighbors or more globally, from the system environment. Temporal events are triggered according to specific schedules specified in a dedicated Modeling_Unit (MUT), which may be affected by some external events too.

Action

An action abstracts a set of treatments upon attributes. It may be activated by the occurrence of an event, or by the explicit invocation of a service. An event may trigger one of several actions. A service execution corresponds to one or several actions.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

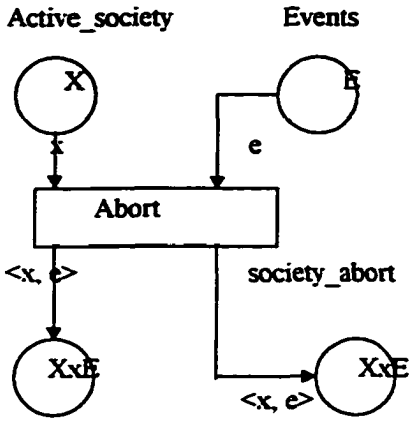


Figure 176:Event and state change

If one considers the event 'society_abort' of fig 176, the event may trigger a service 'create society (x)', to provide a new instance y of the same type X. It may also require a more general service 'abort recovery' which will include the former, and send notifications of abort to the society's server, that may redirect their requests to other instances of the same type.

Message and Service Invocation

The message formalizes Modeling_Units interactions. Interactions correspond to requests for a certain service and associated results, including acknowledgment of receipt and request rejection. An acknowledgment request ensures the effective computation of the required service, while a rejection implies the determination of another instance of Modeling_Unit able to satisfy the request. Information flows are brought within messages. Messages can be grouped under types, to distinguish the different kinds of exchanged information. Each type is characterized by the types of its slots. A slot may contain the sender's identifier, the service name, the list of corresponding inputs and outputs.

All the concepts presented above do not bring much information about the whole system if considered independently in the specification. Thus, one retains to link the two formal models: the structural and the behavioral model.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The structural model provides a coherent description of structural relationships between components, according to an extended entity/relationship model dealing with the *part of* concept. The behavioral model depicts services, attributes and their domains, and triggering links between events and services. The services model depicts communications between Modeling_Units. Any sending or receipt of message triggers an event in the system and may change a Modeling_Unit state. The services model is built applying four steps, for each Modeling_Unit:

- identify the set of possible states.
- identify actions which modify a state.
- identify internal, external and temporal events.
- establish triggering links between events and actions.

This process is not always carried out sequentially. Several heuristics can be applied, depending on the amount of available information on the domain:

- The specifier may determine actions first, and then define the pertinent states of the system as those affected by action executions. The model may be enriched by refining existing states or adding new ones after a check of completeness. This approach is action-driven.

- Another approach in describing the main states of the system and associated event types (i.e., list of possible events occurring at those states). Actions are then determined as executing state changes.

In practice, complex applications use hybrid approaches. This is often due to the lack of information (incomplete requirements), i.e., to the difficulty to express clearly events and states in some components, and to predict them.

5.6.5 Translations Rules into Colored Nets

The life cycle defines the way (i.e. the logic) in which services are executed. The life cycle may be *concurrent*, and its refinement can lead to autonomous, possibly sequential activities. Service refinement is

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

subordinated to *resources* (knowledge) and their concurrent sharing which is more complex than sequential sharing. It requires *mutual exclusion* and *temporal atomicity.*

Therefore, to construct the life cycle net, services are decomposed into observable and non-observable actions, formalized with transitions. Observable actions includes interactions management (i.e., events, messages), services definition and associated (pre and post) conditions. The non-observable part specifies internal state changes, triggering of private (i.e., internal) actions, and eventually, associated returned information.

5.6.6 Modeling_Unit Validation

At a given level of abstraction, the validation of a Modeling_Unit concerns:

- The static information (data types) on which completeness and coherency checks are achieved as for abstract data types. In particular, it is checked whether operations defined on data do not violate their integrity constraints.
- behavioral information handled by associated nets. Qualitative as well as quantitative results are available on nets from various literatures. For instance, it can be found that a certain structure of life cycle leads to deadlock, or that given message can never reach its destination, without having to simulate the net. Furthermore, it can be calculated the mean number of messages (tokens in nets) hold by a Modeling_Unit, or waiting for processing in a given queue (place of the net). Moreover, timed and stochastic nets are also analyzed which provides interesting evaluation of components performances (e.g., mean time of message processing, or of service answering etc.). Globally, the modeling process emphasizes the following properties:

  - coverage of target system activities: through Modeling_Units properties, all system's activities should be covered, even though abstract actions that will be refined.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- behavioral coherency: for cooperative Modeling_Units, the set of associated events and messages should be coherent with respect to their life cycles.

- activities independence: units should cover independent activities of the target system, eventually synchronized with others on shared resources or shared events.

- preservation of dynamic transitions: for a complex service that trigger further actions or services on several components, the atomicity of the whole transition should be guaranteed.

5.6.7 Design of Intelligent Cooperative Societies

5.6.7.1 Knowledge Partitioning Criteria

The underlying system architecture (as described before) is based on the modularization of information representation and analysis of knowledge base societies, by means of the Modeling_Unit formalism. The major advantage is the possibility to specialize information into autonomous modules, to reduce the complexity and inherent time responses in complex problem solving situations.

These advantages are visible both at the level of design and the implementation levels:

Modular design is based on the concept of *knowledge packets*, and the associated control and communication mechanisms, necessary to the cooperation in the system. Knowledge specialization delimits the activation part of the services at their interface according to the object paradigm. This facilitates This facilitates packets validation and reuse, and allow to define the cooperation protocol between them independently from the formalisms they implement.

The complexity of system is reduced when decomposing it into manageable tasks. The control mechanism allows then to detect as early as possible, the part of the system responsible for a sub-problem i.e., *representation and validation* of a certain view- and focus only on this part, called knowledge packet. The cooperation protocol synthesizes the results of sub-problem resolutions, and present it to the end-user interface, relying on the degree of interoperability allowed by the system.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The concept of selection of the adequate formalism dealing with a given sub-system is ensured by the generalization of an artificial intelligence technique, the high-level pattern matching, where two levels are distinguished: the classical pattern matching dealing with simple production rules, and a meta pattern matching whose granularity is not any more the rule, but a packet of rules. This distinction allows to reduce the complexity of a problem solving and mainly, the impact of inferences on a restricted knowledge base i.e., the amount of information necessary to handle a given sub-problem.

From a validation point of view, object-orientation is very promising. However, the possibility of validating autonomous components either by state machines or communicating processes has also shown selected formalism limits, face to the inherent occurrence of modeled entities. Therefore, concurrent formalisms were needed.

5.6.7.2 Description of the Knowledge Schema

The approach consists in structuring an information system as a society of knowledge granules, encapsulated in Modeling_Units, characterized by hierarchical interdependencies, and processed by specific rules. Modeling_Unit properties are mainly described by static properties, facts and/or meta facts, and a behavior expressed dynamically with production rules. Facts and meta facts are exploited by the inference engine, and processed by the production rule language.

Knowledge structure is achieved according to a typology which induces the role of the different modules in a problem resolution process. However, a challenger solution necessitates more than efficient control strategy. It requires the selection of high performance representations of the problem, and efficient paths and conditions.

The difficulty in using declarative knowledge is proportional to their declarative aspects: it is easy to saturate a set of rules to achieve a goal, but it is difficult to predict the amount of attempts the system

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

must make before reaching it. Minimizing these amounts is ensured by defining orders on rules, i.e., define a meta control mechanism on rules.

5.6.7.3 Meta Control Mechanism and Interoperability of Specifications

The control is defined as the selection, at each step of the resolution process, of the action to be performed by the system. It can be located at different levels: local to the knowledge packet, the control is guided by data. The meta control on reasoning is realized by the knowledge base root and is guided by goals. Hereafter, we will focus on the meta control, the local control being classical.

Given a proof to establish (i.e., a property to verify in the target domain), the main role of the meta control is selection of adequate knowledge modules, management of their cooperation, and then synthesis of their partial results. The control strategy is of successive attempts, with chronological backtracking. Different kinds of graph structures and graph search procedures are used. For interdependent domains, it is sometimes necessary to avoid representation redundancy. The control module manages user's request and put them in form so as to create a goal, and to present it to knowledge societies. It decomposes then the goal according to a logical model and elaborates a preliminary planning to satisfy the goal. The planning evolves dynamically, depending on partial results calculated by activated societies. The later are selected by the control according to their ability to resolve a goal (or sub-goal of the initial problem). Eventual conflicts (raising when several societies could be selected for the considered goal) are resolved by choosing the best planning for the actual situation, that is defined by heuristics of the domain.

A proof consists in verifying a property P for a model M, expressed with the formalism F:
Let Sf be the set of formalisms, Sr the set of models and Sp the set of properties to verify on models. A proof Pr is an application:

$$Pr: Sf \times Sm \times Sp \longrightarrow Sf \times Sm \times Sp$$

$$(F, M, P) \longrightarrow (F', M', P')$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

where: F and F' are formalisms, M and M' models, P and P' are properties.

The scheme of the meta proof techniques is given in fig 177.

5.6.7.3.1 Proof in the same formalism

In particular, the proof in the same formalism (i.e., when F=F') consists I four techniques:

a) Model reduction: it reduces the size of the model, with preservation of the searched property. Reduction consists then in abstracting the model to the penitent concepts necessary to establish the proof.

b) Model decomposition: given a property P to prove on a model M, the technique consists in generating a set of sub-models Mi and to apply using the following approaches:
   - prove properties pi on Mis, so that Mis' composition allows to verify P on M.
   - prove initial property P on Mis and deduce that it is verified on the global model by composition.

c) Proof simplification: a proof may be simplified by implication, or equivalence:
   - The former relies on the assertion "if M verifies P', then it verifies P". The proof process for a property P on the model M is then decomposed into three tasks:
     - define a property P' that verifies the assertion
     - prove the property on the model M
     - deduce P from P' on M.

     In this case, Pr is a projection of Sp on Sp' with M and F constant.

     Pr(F, M, P) = (F, M, P').

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

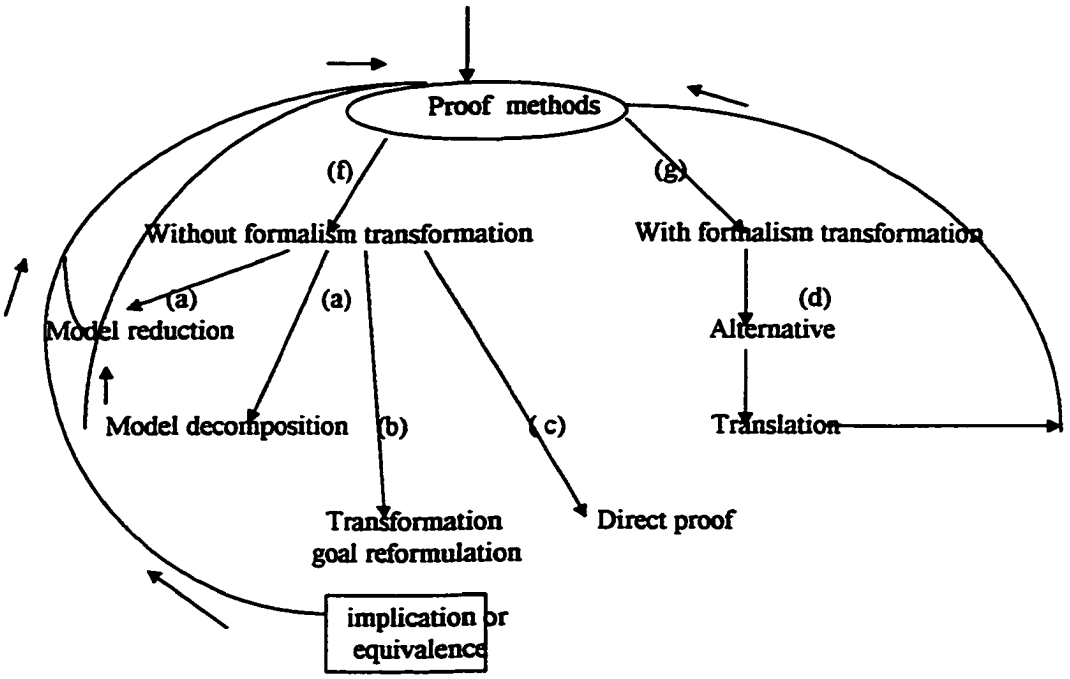


Figure 177:Meta proof techniques

- By equivalence, the assertion "M verifies P if and only if M verifies P" allows to consider the implication part "if M verifies P' then it verifies P" (the other side is not interesting in establishing the proof). Note that the proof by implication can be generalized to a sequence of implications.

d) direct proof: when simplification attempts described above fail, a theorem is applied directly.

The combination, in the same formalism, of the four techniques above allows to elaborate an efficient proof technique. The ordering of these techniques necessitates meta knowledge describing the techniques and their activation pre-conditions, pertinence- and activating them effectively. This role is handled by the super-method defined in the knowledge base root.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

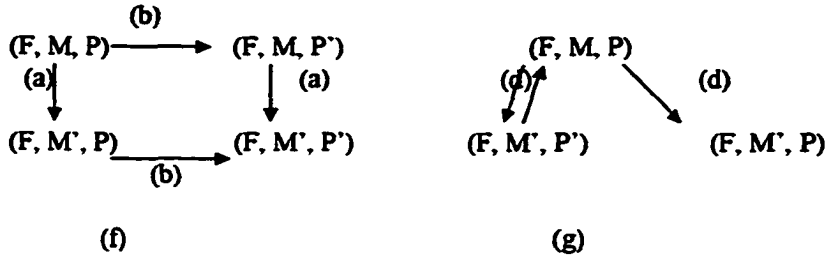


Figure 178:Combining proof techniques

5.6.7.4 Proof in another formalism

When a property cannot be proved in the current formalism, we try to replace the formalism F with an equivalent formalism F' and translate the model M into a model M', while ensuring the condition:

if M' verifies P then "M verifies P" is true

$$(F, M, P) \xrightarrow{Pr} (F', M, P)$$

In this case, Pr is the projection of F on F'.

The meta control is defined in this section provides an efficient strategy to enhance knowledge, formalism and tool cooperation in heterogeneous information systems.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.6.8 Knowledge Base for Representation and analysis of Petri Nets

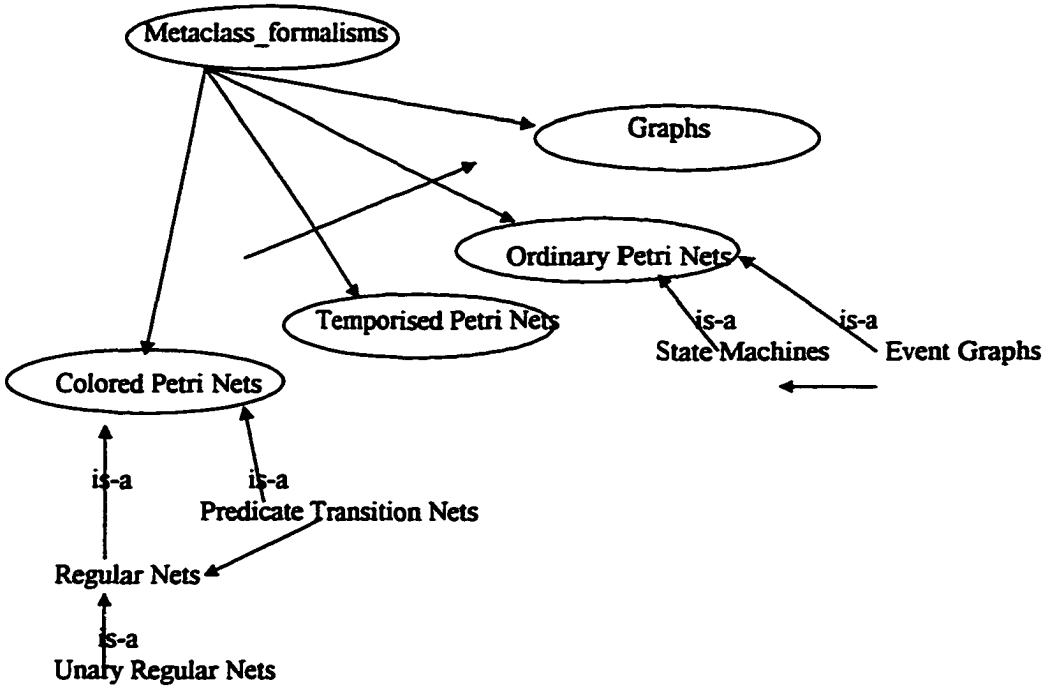


Figure 179:Modeling the world of Petri Net Formalisms

The suggested methodology can be applied to describe the knowledge base of a tool implementing the Petri net formalism world. This world is decomposed into the main Petri nets classes and their associated validation techniques.

Classes are described hierarchically. The path from a level to a lowest one defines the specialization of the class, by attributes enrichment. On the other side, omitting an attribute in the net representation corresponds to going up in the hierarchy. For instance, the class of ordinary net is specialized into the class of temporal Petri nets, by addition of the attribute *time* in the definition of transitions.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Classes cooperate in order to represent, analyze and validate properties of a net. They offer validation techniques associated to the formalism they describe. The meta control plans the proof process, according to the strategy as described before in section 5.6.7.3.

5.6.9 Modular Implementation issues

The impact of the suggested methodology on the design and implementation of intelligent cooperative information systems is discussed here.

At the design phase several features are attractive:

- The design of knowledge modules specialized I the resolution of specific problems and their incremental integration in the knowledge base.

- Concurrent activation of modules to deal with specific tasks, or sequences of activation for complex tasks. The advantage is to be able to reason at the adequate abstraction level which is the task level, to effectuate coherently complex tasks.

- Modules reuse along the same resolution process is possible, by means of a systematic representation transformation. The advantage is to inherit properties of a given model from another model, and to use the tools defined on other representations (other than current representation).

- Knowledge modules cooperate independently from application domains.

At the implementation level, this approach encourages:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Reduction of problem complexity, by decomposition and invocation of the society of modules specialized in solving each sub-problem. It reduces considerably time responses, provided that the resolution process is guided by goals.

- For each problem solving, reduction of the impact of inferences on a restricted knowledge base necessary to the current problem. This reduces, besides time responses, memory allocated to knowledge at a given moment.

- The possibility to focus on a goal, by a backward chaining on necessary sub-goals, which avoid inferring on other goals deductible when using any ward chaining.

Besides, the multi-formalism approach constitutes an attractive step towards physical distribution of knowledge. The structure of Modeling_Units induces several guidelines on the architectural features necessary to support the distribution and the cooperation of the specified system.

Also this approach suggests a validation of knowledge societies from high level specification: the Modeling_Unit life cycle allows to validate the role of each society and to predict its possible behaviors. On the other side, the control either local to societies, or global cam be validated. The ' part of ' relationship defined on the underlying model allows to consider variable granularities of knowledge which makes possible reasoning at different abstraction levels.

In general, the involvement of large knowledge, shared or exchanged between several components, while achieving a task needs efficient control, to manage multiple events activating modules cooperation. At higher abstraction level, knowledge must have ability to deal with inherent complexities of processes: non-determinism, parallelism, back-tracking, hierarchical activity structure and external human intervention. A meta-control strategy based on a hierarchical representation of knowledge is highly suitable to deal with this complexity.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.6.10 Introduction to Inter-schema knowledge-based system

In a distributed system, it is very natural to access databases across nodes which may be supported by different schemas. These schemas may have different views but still the question is if one can propose a logic-based language to express interdependencies between classes belonging to different schemas. For simplicity, one may assume all the different schemas are based upon the same data model, which is entity-relationship model. The purpose is to propose a logic-based language to express interdependencies between classes belonging to different schemas. Such interdependencies will allow a designer of a set of cooperative information systems to establish several relationships between both the intensional definition and the set of instances of classes represented in different schemas. For example, one can assert in our language that the concept represented by the class *Graduate Student* in the schema S_1 is the same as the concept represented by *Senior Student* in S_2. Such assertion implies a sort of intensional equivalence between the two classes, but does not imply that the set of instances of the former is always the same as the set of instances of the latter. On the other hand, one can assert that the set of instances of the class *Teacher* in the schema S_2 is always disjoint from the set of instances of the class *Tutor* in S_3, even if there exists a form if intensional inclusion between the latter and the former.

Once we have a set of assertions of the above mentioned kinds, we may benefit in several ways from the possibility of reasoning about them. These inference mechanisms will allow reasoning to be carried out. Based on the formal semantics of the language, we can indeed devise suitable reasoning procedures that allow one to draw useful inferences on interschema knowledge. For example, one can check whether one class represented in the cooperative information systems is coherent, i.e. it has an empty extension in every state of the databases, or can deduce that the extension of a class A in the schema S_i is always a subset of the extension of the class B in S_j, so that accessing S_i is useless if we want to anyway retrieve all the instances of the concept represented by B.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.6.10.1 Basics of Entity-relationship model

The conceptual model used here is assumed to be entity-relationship model. As usual, the universe of discourse of discourse is divided into two levels, called the *instance level* and the *class level*. The basic elements of the *instance level* are objects, i.e. atomic things that can be identified using a unique symbol. Such a symbol plays the role of object identifier. These objects are grouped into classes, and the objects belonging to a class C are called the *instances of* C.

Objects can be aggregated into *tuples*. The number of objects which are components of a tuple is called the arity of the tuple. Tuples are grouped into sets that are called relationships. The tuples belonging to a relationship R are called the *instances* of R. Each relationship R has an associated arity, which is also the arity of the tuples that are instances of R. Such an arity represents the number of objects that are aggregated in every tuple that is an instance of R.

The *class level*, which is specified in terms of a so-called *schema*, is constituted by an alphabet of class symbols, and by the specification of how the classes are related to each other. The *alphabet* is simply a list of symbols partitioned into entities, relationships, roles, attributes and domains. An *entity* is an abstraction for a class of objects or instances, denoted by a name. The properties of an entity are modeled in terms of a set of attributes and a set of relationships with other entities. Entities are organized in hierarchies, specified by means of a binary relation over entities, called ISA relation. The fact that the entity E is related to the entity F by means of the ISA relation in a schema S specifies that the set of instances of E is a subset of the set of instances of F in every database state corresponding to S.

A *relationship* is a class of tuples of objects of the same arity, described by a name, a set of roles and a set of entities, one for each role. A role denotes a component of the relationship. A relationship has as many roles as the arity of the tuples it represents. For example, the relationship *Marriage*, of arity 2, has two associated roles, *husband* and *wife*. The fact that E is the entity associated with the role U of the relationship R means that in every tuple r of R, the U-component of r is an instance of E.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A *domain* is a set of values. We assume that a suitable set of constructs such as enumeration, interval, basic domains as integer, string etc. in the style of type definitions in conventional programming languages are available for specifying the structure of domains. An *attribute* is a relation between an entity or a relationship, and a domain.

5.6.10.2 Specification of interschema knowledge

Suppose that $S_1, S_2, \ldots, S_n$ are the schemas of n cooperative information systems and let $A_1, A_2, \ldots, A_n$ their alphabets. Since we want to keep separated the symbols of the various schemas, we assume that for each $i \in \{1 \ldots n\}$, the elements of A_i are subscripted with i. Thus, for example, $Teacher_2$ denotes a class Teacher in schema 2. Our objective is to devise a method for specifying interschema knowledge in terms of interdependencies between classes belonging to different schemas. It is obvious that in order to do so, we need a mechanism for denoting the classes of a schema. To this end, we make use of a language for the specification of entity expressions and relationship expressions. Roughly speaking, an S_i-entity expression is an expression denoting a set of objects, formed by using special constructors applied to the classes explicitly represented in the schema S_i. For example, if $Student_1$ and $Woman_1$ are entities of S_1, then $(Student_1 \cap Woman_1)$ is an S_1-entity expression denoting the set of objects that are instances of both entities. Note that the symbols used in an S_i-expression must belong to the alphabet A_i of S_i. Analogous considerations hold for relationship and attribute expressions.

The syntax and the semantics of class expressions are described below. Using the class expressions, inter-schema knowledge is specified by means of four different types of assertions, that are used to state semantic interdependencies among the classes of the various schemas, and are discussed later.

5.6.10.2.1 Class expressions

Here we refer to a single generic schema S_i, and describe the language to form S_i-class expressions. There are three kinds of class expressions, namely, entity expressions, relationships expressions, and attribute expressions. Let us discuss entity expressions first. The simplest example of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

entity expression is the one formed by the name of a single entity. More general concepts are described by means of suitable constructors that allow for building complex expressions.

The characteristics of the language we use to form entity expressions resemble those of the so-called frame-based languages in AI. An entity expression is formed by means of the following rules:

$$
\begin{aligned}
C, F \rightarrow\ & \top \mid \\
& \bot \mid \\
& E \mid \\
& C \cap F \mid \\
& C \cup F \mid \\
& \neg C \mid \\
& (\forall R[U].T_1: C_1, \ldots, T_n: C_n) \mid \\
& (\exists R[U].T_1: C_1, \ldots, T_n: C_n) \mid \\
& \forall A.D \mid \\
& \exists A.D
\end{aligned}
$$

$$
\begin{aligned}
D \rightarrow\ & V \mid \\
& \{v_1, \ldots, v_n\}
\end{aligned}
$$

where the following meta-variables are used:

- C and F are entity expressions, and D is a domain.
- E is a name of an entity, R is a name of a relationship, $T_1, \ldots, T_n$, U are names of roles, A is a name of an attribute, and V is a name of a domain.
- R[U] denotes the projection of R onto the role U (i.e., the set of objects that are U-components of the tuples of R).
- $v_1, v_2, \ldots, v_n$ are values belonging to domains.

The intuitive meaning of entity expressions is as follows:

- $\top$ and $\bot$ represent the universal entity (the entity whose instances are all the objects in the schema) and the empty entity, respectively.
- E denotes the set of instances of the entity E.
- $C \cap F$ represents the set of objects that are instances of both C and F.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- $C \cup F$ represents the set of objects that are instances of C or F.
- $\neg C$ represents the set of objects that are not instances of entity C.
- $(\forall R[U].T_1 : C_1, \ldots, T_n : C_n)$ represents the set of objects x such that for each tuple r of R with x as U-component, the T_i-component of r belongs to the extension of C_i.
- $(\exists R[U].T_1 : C_1, \ldots, T_n : C_n)$ represents the set of objects x such that there exists a tuple r of R with x as U-component and y as T_i-component, such that y belongs to the extension of C_i.
- $\forall A.D$ represents the set of objects that are associated with values in D by attribute A.
- $\exists A.D$ represents the set of objects that are associated with at least one value in D by attribute A.

The formal semantics of entity expressions is based on the notion of interpretation. An *interpretation I* $= (\Delta^I, \gamma^I, .^I)$ for S_i consists of:

- a set of Δ^I (the *universe* of I, which is assumed to include all the elements of the domains used in the schema),
- a function γ^I that maps every value to an element of Δ^I in such a way that $\gamma^I(a) \neq \gamma^I(b)$ if $a \neq b$, and
- a function $.^I$ (the interpretation function of I) that maps every entity expression to a subset of Δ^I, every attribute to a subset of $\Delta^I \times \Delta^I$, and every relationship to a set of labeled tuples over Δ^I. In particular, if R is a relationship whose set of associated roles is rol(R) = { $U_1, U_2, \ldots U_m$}, then R^I is a set of labeled tuples of the form ($U_1 : u_1, \ldots, U_m : u_m$), where $u_1, \ldots, u_m \in \Delta^I$ (we write r[U] to denote the value associated with the U-component of the tuple r). Moreover, the following equations are satisfied by I:
- $T^I = \Delta^I$
- $\bot^I = \varnothing$
- $\{v_1, \ldots, v_n\}^I = \{\gamma^I(v_1), \ldots, \gamma^I(v_n)\}$
- $(C \cap F)^I = C^I \cap F^I$
- $(C \cup F)^I = C^I \cup F^I$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- $(\neg C)^I = \{a \in \Delta^I \mid a \notin C^I\}$
- $(\forall R[U].T_1:C_1, \ldots, T_n:C_n)^I = \{a \in \Delta^I \mid \forall r \in R^I.\ r[U] = a \Rightarrow (r[T_1] \in C_1^I \wedge \ldots \wedge r[T_n] \in C_n^I)\}$
- $(\exists R[U].T_1: C_1, \ldots, T_n:C_n)^I = \{a \in \Delta^I \mid$

  $\exists r \in R^I.\ r[U] = a \wedge r[T_1] \in C_1^I \wedge \ldots \wedge r[T_n] \in C_n^I\}$
- $(\forall A.V)^I = \{a \in \Delta^I \mid \forall (a, b) \in A^I.\ B \in V^I\}$
- $(\exists A.V)^I = \{a \in \Delta^I \mid \exists (a, b) \in A^I.\ b \in V^I\}$

Let us now turn to the relationship expressions. A relationship expression is simply an expression of the form:

$R[U_1, U_2, \ldots, U_n]$

where R is a relationship name, and $U_1, U_2, \ldots, U_n$ are the roles associated to R. The semantics of the above expression is simply given by:
$(R[U_1, \ldots, U_n])^I = R^I$ where I is an interpretation.

Finally an attribute expression is simply constituted by an attribute name A, whose semantics in an interpretation I is simply A^I. From now on, we will use the term *class expression* to mean either an entity expression or a relationship expression, or an attribute expression, formed according to the syntax rules presented above.

Note that the notion of interpretation corresponds to the notion of database state. Notice also that in specifying the semantics of class expressions we have implicitly provided the formal semantics for entities, relationships, roles, attributes and domains in the schema S_i.

5.6.10.2.2 Interschema Assertions

Suppose we are dealing with n cooperative information systems, whose schemas are $S_1, S_2, \ldots, S_n$, respectively. Let A_i denote the alphabet of the schema S_i, for each $i \in \{1, \ldots, n\}$. Also, let S_0 be another

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

schema with alphabet A_0, called the *common knowledge schema* of the cooperative information systems. Intuitively, S_0 represents the properties of the entities, the relationships and the attributes that should be considered common knowledge in the cooperative information systems.

Interschema knowledge is specified in terms of interschema assertions. There are four kinds of assertions, whose forms are:

$$C \equiv_{int} F$$

$$C \leq_{int} F$$

$$C \equiv_{ext} F$$

$$C \leq_{ext} F$$

where in every assertion, C represents an S_i-class expression, and F represents an S_j-class expression, in such a way that C and F are class expressions of the same types (either class expressions or relations expressions or attribute expressions), and either $i \neq j$, or $I = j = 0$. In other words, an interschema assertion represents either an interdependency between two class expressions belonging to two different schema, or an interdependency between two class expressions of the common knowledge schema. Moreover, if C and F are relationships expressions, there is the constraint that the set of roles appearing in C has the same cardinality as the set of roles appearing in F.

The intuitive meaning of the four kinds of assertions presented above will be discussed now and then the formal semantics will be elaborated. For the sake of simplicity, in the observations on the intuitive meaning of assertions, we only refer to assertions on entities.

The first assertion states that the entity expression C is intensionally equivalent to F. This means that although the extension of C may be different from the extension of F, the concept represented by C is in fact the same as the concept represented by F. For example, $UndergraduateStudent_1$ is intensionally equivalent to $(Student_2 \cap \neg\ GraduateStudent_2)$, in the sense that, even if the instances of the two

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

expressions may be different in a particular state of the cooperative information systems, the concept of *UndergraduateStudent* in the schema S_1 is fully captured by the above entity expression in the schema S_2.

The second assertion states that the entity expression C (say of schema S_i) is intensionally less general than the entity expression F (of schema S_j). This means that there is a sort of containment between C and F, and this containment is conceptual, not necessarily being reflected at the instance level. In other words, the above intensional relationship implies that, if S_i and S_j referred to a unique set of objects in the real world, then the extension of C would be a subset of F. For example, $Tutor_2$ may be declared intensionally less general than $Teacher_3$, because the concept of tutor is subsumed by the concept of teacher.

The third assertion states that the extensions of C and F are always equivalent. In this case we are asserting that in every state of the cooperative information systems, the set of objects that are instances of C in S_i is the same as the set of objects that are instances of F in S_j.

The fourth assertion states that the extension of C is always a subset of the extension of F. For example, if the schema S_1 refers to a University Department, and the schema S_2 refers to the whole University, then the extension of $Student_1$ is always a subset of $Student_2$.

In the light of the above discussion on the intuitive semantics of the assertions, we can point out some basic observations on their use in the specification of interschema knowledge.

Notice first of all that, in the case where the assertion relates two S_0-class expressions, the assertion specifies a piece of knowledge about the world represented in the various information systems. The set of such assertions are intended to specify the knowledge that is independent of the views of the world as reflected in the schemas $S_1, \ldots, S_n$. Obviously, in those applications wheresuch a knowledge is not available, the set of assertions relating S_0-class expressions will be empty.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

On the other hand, the assertions relating an entity expression relative to a schema different from S_0 to an S_0-entity expression specify a set of relationships between the classes represented in the various schemas and the class represented in the common knowledge schema. Notice that, if naming conflicts between the schema S_i and S_0 have already been solved, one can assume that an assertion of the form

$$E_i^h \equiv_{int} E_0^h$$

is specified for every entity name E_i^h of the schema S_i.

Finally, the assertions relating two entity expressions belonging to two different schemas are used to establish several conceptual relationships between the classes belonging to different schemas.

In order to assign the formal semantics to assertions, we introduce the notion of global information system, and the notion of interpretation for it. A global information system is intended to represent a collection of cooperative information systems. At the class level, a global information system G $= < S_0, S_1, \ldots, S_n, \Sigma >$ is constituted by

- the collection of the n schemas $S_1, S_2, \ldots, S_n$ of the various information systems
- a common knowledge schema S_0
- a set Σ of assertions of the forms presented above.

An interpretation I for $G = < S_0, S_1, \ldots, S_n, \Sigma >$ is built out starting from n interpretations $I_1, I_2, \ldots, I_n$ for $S_1, \ldots, S_n$, respectively. In particular, I is a triple $I = (\Delta^I, \gamma^I, .^I)$ such that Δ^I is the union of the various Δ_i^I ($I \in \{1 \ldots n\}$), and γ^I and $.^I$ are extensions of the union of the various γ_i^I and $._i^I$ ($i \in \{1 \ldots n\}$), respectively, in such a way that every symbol in S_0 is given an extension by I.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

We assume that in the assertion $C_1 \equiv_\beta C_2$, where β is either int or ext, C_1 is an S_i-class expression and C_2 is an S_j-class expression.

- The assertion

$$C \equiv_{int} F$$

where C and F are entity expressions, is satisfied by I if for every $a \in C^I$ if and only if for every $a \in \Delta_i^I \cap \Delta_j^I$, $a \in C^I$ if and only if $a \in F^I$.

The assertion

$$P[U_1, \ldots, U_n] \equiv_{int} R[T_1, T_2, \ldots, T_n]$$

where P and R are relationships, is satisfied by I if for every $u_1, \ldots, u_n \in \Delta_i^I \cap \Delta_j^I$, $< U_1: u_1, \ldots, U_n: u_n > \in P^I$ if and only if $< T_1: u_1, \ldots, T_n: u_n > \in R^I$.

- The assertion

$$C \leq_{int} F$$

where C and F are entity expressions, is satisfied by I if for every $a \in \Delta_i^I \cap \Delta_j^I$, $a \in C^I$ implies $a \in F^I$. The assertion

$$P[U_1, \ldots, U_n] \leq_{int} R[T_1, \ldots, T_n]$$

where P and R are relationships, is satisfied by I if for every $u_1, \ldots, u_n \in \Delta_i^I \cap \Delta_j^I$, $<U_1: u_1, \ldots, U_n: u_n > \in P^I$ implies $< T_1: u_1, \ldots, T_n: u_n> \in R^I$.

- The assertion

$$C \equiv_{ext} F$$

where C and F are entity expressions, is satisfied by I if for every $a \in \Delta^I$, $a \in C^I$ if and only if $a \in F^I$. The assertion

$$P[U_1, \ldots, U_n] \equiv_{ext} R[T_1, T_2, \ldots, T_n]$$

where P and R are relationships, is satisfied by I if for every $u_1, \ldots, u_n \in \Delta^I$,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$< U_1: u_1, \ldots, U_n: u_n > \in P^I$ if and only if $< T_1: u_1, \ldots, T_n: u_n > \in R^I$.

- The assertion

$$C \leq_{ext} F$$

where C and F are entity expressions, is satisfied by I if for every $a \in \Delta^I$, $a \in C^I$ implies $a \in F^I$

The assertion

$$P[U_1, \ldots, U_n] \equiv_{ext} R[T_1, \ldots, T_n]$$

where P and R are relationships, is satisfied by I if for every $u_1, \ldots, u_n \in \Delta^I$,

$< U_1: u_1, \ldots, U_n: u_n > \in P^I$ implies $< T_1: u_1, \ldots, T_n: u_n > \in R^I$.

Analogous definitions hold for the semantics of assertions on attribute expressions.

An interpretation I for $G = < S_0, S_1, \ldots, S_n, \Sigma >$ is called a model of G if every assertion in Σ is satisfied by I. If σ is an assertion, one can say that σ is a logical consequence of G (written as $G \models \sigma$) if σ is satisfied by every model of G.

Now using the reasoning techniques, the interschema assertions can be used to make inferences on the knowledge represented I the cooperative information systems. The method consists of associating a global information system $G = < S_0, S_1, \ldots, S_n, \Sigma >$ a so-called interschema knowledge network, which is a sort of semantic network representing the relevant knowledge about the relationships holding between the classes of the schemas. More precisely, the interschema knowledge network associated with G is a graph whose nodes are the entities of the various schemas, and whose arcs are partitioned into two categories. ISA arcs and DISJOINT arcs. An ISA arc from A to B represents the fact that in every model of G the extension of A is a subset of the extension of B. A DISJOINT arc connecting A and B represents the fact that in every model of G the extensions of A and B are disjoint.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.7 Components, Algorithms & Procedures

As described briefly before, the proposed information retrieval system is designed to have four main layers of software of which some layers will run on various nodes in a distributed environment and some will run on the local node where the query is initiated with. The details of the functionalities of the various software components involved in the system, their associated knowledge representations, types of knowledge domains required , control flows with examples have been treated below and also the formal algorithms have been added thereon. The system basically has a limited world-knowledge in the domain of academic information retrieval on which it depends very heavily. Also it tries to pursue as much accurate syntactic analysis as possible. Various statistical methods, e.g. based on word co-occurrences and mutual information are likely to generate very high error rate when it comes to phrase searching. This system instead searches for the concept structure.

This system is novel for its distributed negotiation methods, knowledge-based analysis, concept processing instead of simple syntactic matching or following a purely statistical models, logic-based analysis. This system when in action, will accept queries from the users in simple natural language and will use knowledge-bases for classification and concept building and finally will express the query in concepts. The knowledge-bases have been elaborated in great details and precisely.

The concept matching that has been used is based upon semantics and is independent of syntax or even languages. Even pushing it to the extreme, it offers the hope of retrieving information in multiple languages or even multi-media based documents where an image or an audio clip or written text can be a source of information. The logic-based processing contains the rules that need to be applied for derivations. More and more rules can be added to bolster the capacity of the system in future.

5.7.1 Overview

The query-retrieval system runs in conjunction with a knowledge-base retrieval system which maintains the world-knowledge in the domain of educational information retrieval and answers queries. The format of the knowledge-base is knowledge-frame using slots and relations. This particular knowledge is used at

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

many levels in the rest of the system at linguistic analysis, generalization etc.. An abstract view of the architecture can be seen in fig 179.

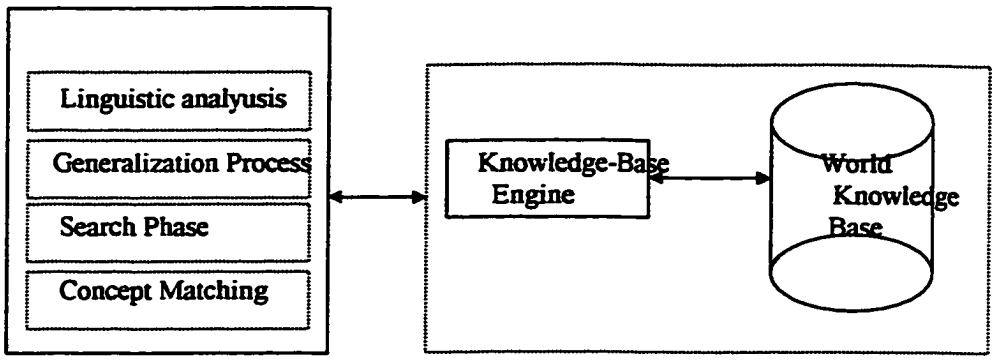


Figure 180:Broad overview of the system

All the queries will be routed through KB-engine. The query retrieval system has many phases.

5.7.2 Functional components

5.7.2.1 World Knowledge-base

The common knowledge of the world is classified into three different types on a major categorization: 1) The entities which are involved with the query language and their attributes (language-entities), 2) the class hierarchy and the relationships among the classes (class knowledge), 3) record of various items, their attributes and relationships (encyclopedia).

The language-entities set is made of frames which are in turn made of many filled slots or relations. Each slot can be a value or a pointer to another frame. The frames represent each important entity. Each relation can have an opposite and reverse relation. There is a frame for one unfilled frame and one for each instantiation. The entities involved with query language are *Author, Topic, Subject, Year_Of_Publishing, Publisher, Related subjects, Title, Language, Rating.* The frame for entity Author, Topic, Subject, Year of Publishing, Related Subjects, Title, Language, Rating etc. are shown in fig 181.

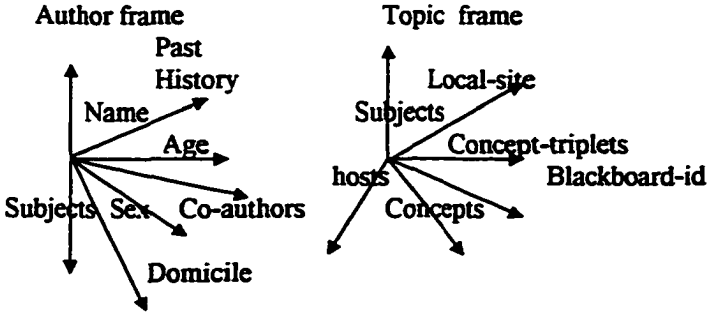
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The class hierarchy is a combination of semantic network and production rules that represents the various types of topics (rather the singular concepts) and the various relations with the other concepts. The relationships are mainly on the basis of concepts containing the generalizations, opposites, synonym, antonym, hyponym, meronym (part of another concept), holonym (superset of another concept). But all these relationships make sense once the dimension has been determined by meaning. As a concept or an abstract word can be used to mean various things, so the meaning is determined first. Then the related class hierarchy is looked up. The meanings are determined by the linguistic role of the concept (i.e. the relationship of the concept with other words) and presence of any identifiable concepts which are closely related to the given concept.

For example, *Decline* can mean a geometrical downward slant, or *Decline* can mean a refusal or *Decline* can mean an avoidance within a series of activities.

Once the meaning is determined, then the relationships with other concepts makes a meaning. It started with the need for representation of topic classes, determining their corresponding concepts and arriving at the subject from these. For example,

In a query where main topic is "*Second world war*", it can be determined from the dictionary of world knowledge that this is an instance of war fought in history. So the concept (which is "<wars in past>: second world war") will be "wars in past: modern" , more generally "wars in history", more generally "history".



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

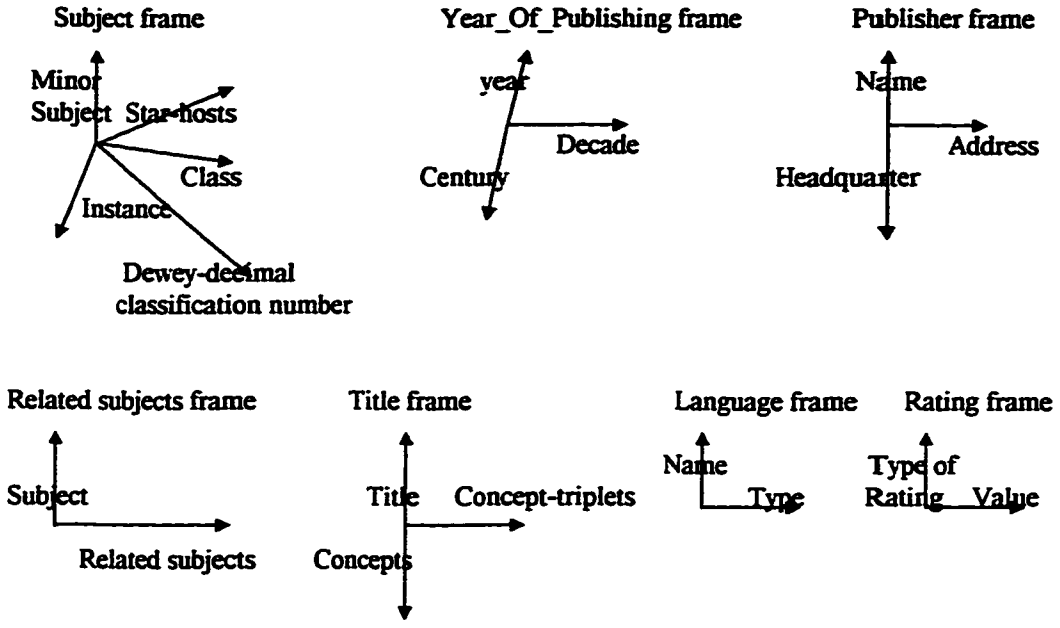


Figure 181:Examples of unfilled frames

The semantic net of the concepts is shown in fig 182. In general many various relations can be established using the directed graphs involving concepts or even variable nodes. The semantic nets have been extended further by 'Kowalski clausal form' thus making it equivalent to first-order predicate logic. There the consequents are represented by solid lines and the antecedents are represented by dotted lines. Even the semantic nets can be partitioned into spaces corresponding to clauses. The number in the relationship tag refers to the sequence number of meaning.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

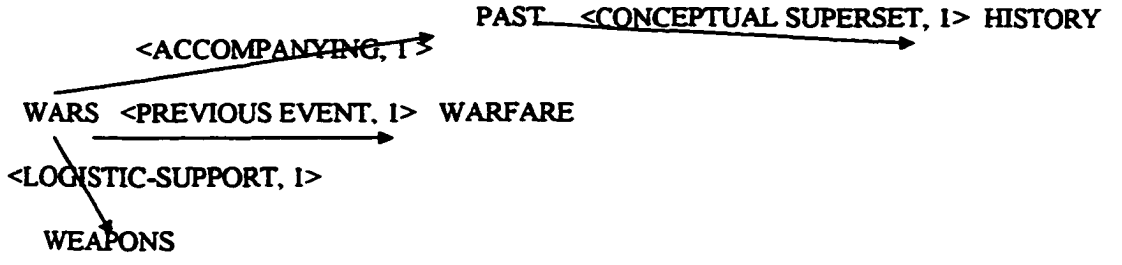


Figure 182:Example semantic net for concept classes

The encyclopedia is the third knowledge-base which behaves like a dictionary. For every word, this dictionary will be the first to look up. Each word from the world-domain is entered in the encyclopedia as on a linguistic frame (or case frame) (which is further indexed by the word or word concept from Roget's International Thesaurus (RIT) that covers the context and linguistic structure of the clause and along with it it are written the meaning concept, closely related concepts, the context etc. in simple text. Once a match is found with the word (rather the concept) and the linguistic frame matches, a small natural language parser engine can fire off to convert the meaning text into a semantic net. The "ISA" relationship gives the generalization of the concept, others give the adjuncts or substitutes. Using these substitutes and closely related words and the entry index, the subject indexor tries to move on the class hierarchy graph trying to move high up in generality so that the subject index in Dewey's Decimal system is reached.

This knowledge-base will be used for purposes other than subject generalization also. The various methods offered will be as:

Lookup_Encyclopedia_Frames(),

Lookup_Encyclopedia(),

Find_Unfilled_Frame(),

Find_Filled_Frame(),

Find_Class_Hierarchy()

Besides, additional knowledge to the various subsystems as the encyclopedia, the class hierarchy or the query-attribute frames can be added using extra methods.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.7.2.2 Linguistic layer:

This is the foremost layer in the system accepting user queries written in a subset of natural language. This layer accepts the query and does some preliminary processing using linguistic analysis. In this preliminary stage the query parsing happens in three phases:

- 1) Parsing the query attributes,
- 2) Adding the subject tag & determining the subject,
- 3) Attaching important semantic structural tags in the topic.

1) Parsing the query attributes.

This top-level parsing simply figures out all the attributes from the query (as author, year, subject, title, topic, publisher etc.). A parse tree is built and the query is expressed in a language based on relational algebra (e.g. SQL). A typical parse tree is shown below in fig 183 where attribute names are in tags and values are without tags.

Example: Get text on performances of Confederate army in American Civil war which was written in 1920s from South America.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

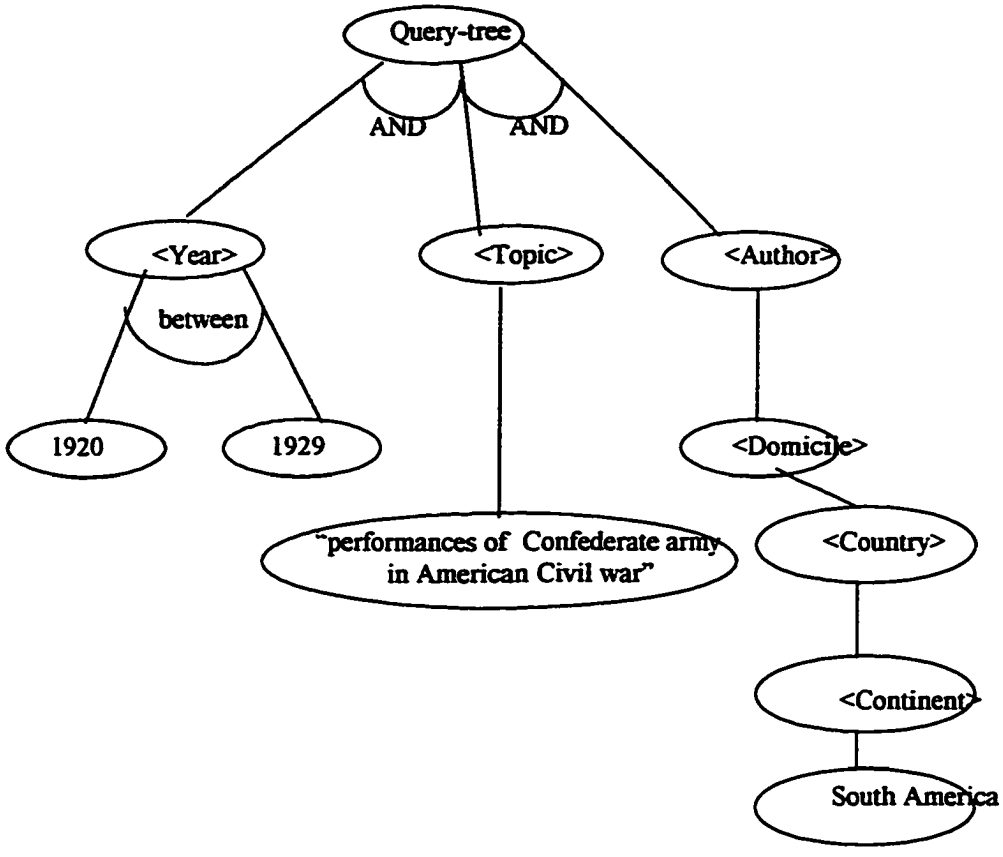


Figure 183:Parse tree for a typical query

This is very light scan of the text being equivalent to a lexical analysis where the keywords are searched. All the keywords are kept in a keyword folder where each keyword has its set of synonyms and a small knowledge base . the lexical analyzer looking for the attributes and the predicates for them.

The and then it digs further into the topic statements and for each important term, clause or sentence it generates some tags indicating the subjects those clauses can be classified as. Along with these subject tags, also some broad classes are identified as temporality, geographical location, societal information, races, category and these are represented as additional subject tags following the main subject tag. The subject is written as :

< Class>: <Derived Class>: <Derived Class>:...:Instance <Derived Class>:...:Instance

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

An elaborate example is given here:

Query Topic: "Deepest Snow on the Alps mountain in Southern Europe".

In here, the primary head in the clause is "Snow" which is part of nature. The other adjunct is "Deepest" which qualifies the head-noun. The other modifier is "Alps mountain in Southern Europe". Again within this modifier sub-clause, the head is "Alps mountain" which can be represented as "class : mountain, instance: Alps" and mountain again belongs to nature. So there is a repeat of the subject 'nature' which makes it more pronounced in the clause. The modifier is "Southern Europe", where again the head is "Europe" (the qualifier being "Southern") which is a continent which belongs to geography (which is part of nature). Generalization stops here because the subject hierarchy (which may be Dewey's decimal hierarchy or Library of Congress hierarchy) has been reached. In this example, the chief subject (due to its repeats) was "Nature" , "Snow" and the second specific subject was "geography: Southern Europe". Thus,

Subject: <Nature>:<Snow>: <Geography: Southern Europe>

A subject is selected to be chief than others depending on its number of occurrences, if it directly follows from the head subject or head-verb. This is some type of an extension of van Dijk's model [101].

5.7.2.2.1 Fast Parsing with TTP Parser

TTP (Tagged Text Parser) is a fast text parser based on the Linguistic String Grammar which was developed by Sager [102]. This parser currently contains 400 grammar productions, but it is by no means complete. The parser's output is a regularized parse tree representation of each sentence, that is, a representation that reflects the sentence's logical predicate-argument structure. For example, logical subject and logical object are identified in both active and passive sentences, and noun phrases are organized around their head elements. The parser is equipped with a powerful skip-and-fit recovery mechanism that allows it to operate effectively.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

TTP is a full grammar parser, and initially, it attempts to generate a complete analysis for each sentence. However, unlike an ordinary parser, it has a built-in timer which regulates the amount of time allowed for parsing one sentence. If a parse is not returned before the allotted time elapses, the parser enters the skip-and-fit mode, when it tries to forcibly reduce incomplete constituents, possibly skipping portions of input in order to restart processing at a next unattempted constituent. The result of this strategy is an approximate parse, partially fitted using top-down predictions. The fragments skipped in the first parse are not thrown out, instead they are analyzed by a simple phrasal parser that looks for noun phrases and relative clauses and then attaches the recovered material to the main parse structure.

Head-Modifier Structures

The syntactic phrases extracted from TTP parse trees are head-modifier pairs. The head in such a pair is a central element of a phrase (main verb, main noun etc.), while the modifier is one of the adjunct arguments of the head. Take a specific example from the Wall Street Journal database:

The former Soviet president has been a local hero ever since a Russian tank invaded Wisconsin.

The tagged sentence is given below, followed by the regularized parse structure generated by TTP.

The/dt former/adj Soviet/adj president/nn has/vbz been/vbn a/dt local/adj hero/nn ever/adv since/int a/dt Russian/adj tank/nn invaded/vbd Wisconsin/np_/per

The above tagged sentence shows the parts-of-speech added to each word of the sentence. Now the predicate-argument parse structure is given.

```
[assert
 [[perf[HAVE]]
   [[verb [BE]]
     [subject
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
[np
 [n PRESIDENT]
 [t_pos THE]
 [adj [FORMER]]
 [ adj [SOVIET]]]]

[object
 [np
  [n HERO]
  [t_pos A]
  [adj [LOCAL]]]]

[adv EVER]
[sub_ord
 [SINCE
  [[ verb [INVADE]]
   [subject
    [np
     [n TANK]
     [t_pos A]
     [adj [RUSSIAN]]]]
   [object
    [np
     [name [WISCONSIN]]]]]]]]]
```

Figure 184:Predicate-argument parse structure

It should be noted that the parser's output is a predicate-argument structure centered around main elements of various phrases. If fig 184, BE is the main predicate (modified by HAVE) with 2 arguments (subject, object) and 2 adjuncts (adv, sub_ord). INVADE is the predicate in the subordinate clause with 2 arguments (subject, object). The subject of BE is a noun phrase with PRESIDENT as the head element, two modifiers (FORMER, SOVIET) and a determiner (THE). From this structure, one can extract head-modifier pairs that become candidates for compound terms. The following types of pairs are considered: 1) a head noun and its left adjective or noun adjunct, 2) a head noun and the head of its right adjunct, 3) the main verb of a clause and the head of its object phrase, and 4) the head of the subject phrase and the main verb. These types of pairs account for most of the syntactic variants for relating two words (or simple phrases) into pairs carrying compatible semantic content. For example, the pair *retrieve+information* will be extracted from any of the following fragments: *information retrieval system; retrieval of information from databases; and information that can be retrieved by a user-controlled interactive search process.* In the example, the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

following head-modifier pairs are extracted (pairs containing low-content elements, such as BE and FORMER, or names as WISCONSIN will be later discarded):

PRESIDENT+BE, PRESIDENT+FORMER, PRESIDENT+SOVIET, BE+HERO, HERO+LOCAL, TANK+INVADE, TANK+RUSSIAN, INVADE+WISCONSIN

It may be noted that three-word phrase *former Soviet president* has been broken into two pairs *former president* and *Soviet president*, both of which have different meanings from the original phrase and this may have negative effect on retrieval precision. Thus a longer phrase may be more appropriate. Also while generating compound terms, one needs to identify the negative terms, that is, those whose denotations have been explicitly excluded by negation. Even though matching of negative terms was not used in retrieval, one can easily prevent matching a negative term in a query with its positive counterpart in the database by removing known negative terms from queries.

5.7.2.2.2 Proper Noun Interpreter

In the course of selecting the terms to be matched, one important candidate is the particular proper nouns (as 'Argentina', 'FAA' etc.) as well as the 'category' level proper nouns (e.g. third-world country, government agency) would improve precision performance. The proper noun interpreter provides a canonical representation of each proper noun; a classification of each proper noun into one of thirty-five different categories (as temporality, geography, societal information, races, category, nature, part-of, superset-of, previous-event, future consequence, verbal reaction, evaluation, metaphor, event, work, official, physical, conceptual, intellectual, emotion, moral, brutal, peace, move, carry, give, insert, withdraw, take, fight, think, educate, learn, mean, free), a means for expanding group nouns into their constituent members. These nouns result in relation-concept representation which will be used in concept-graph matching as explained later.

5.7.2.2.3 Complex Nominal Phraser

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The complex nominal phrases is another category that contributes in substantive content. This are the noun + noun, noun + noun + noun or adjective + noun types of combinations in any clause (specially the main clause) and they offer a linguistic means for precise conceptual matching, as do proper nouns. Examples of these nominals are "debt reduction", "government assistance", "health hazards" etc.. The conceptual content of complex nominals can be expressed in synonymous phrases, in a different way than can the conceptual content of proper nouns. The complex nominals can be used in two levels: once for subject resolution and once as part of matching arguments. But the handling methods are different in these two cases.

Once having recognized all the CN phrases, the substitutable phrases for each complex nominal are found by computationally determining the overlap of synonymous terms suggested by RIT and statistical corpus analysis. These processes serve to identify all second order associations between each complex nominal constituent and the terms in the index set. Second order associations exist between terms that are used interchangeably in certain contexts. The premise here is that, if for example, terms a and b are both frequently premodified by the same set of terms in a corpus, it is likely that terms a and b are substitutable for each other within these phrases. The use of both corpus and RIT information appears to limit the over-generation that frequently results from automatic term expansion.

The terms that exhibit second order associations are compiled into equivalence classes. These equivalence classes provide substitutable synonymous phrases for the detected complex nominals and are used by the matching algorithms in the same manner that the original complex nominals are used. The complex nominals and their substitutes are first used in the upstream matching of query to documents and will be explained later.

In addition, each complex nominal and its assigned relation provides a Concept-Relation-Concept triplet to the relation concept detector module for the final round of matching. For that module, semantic relations between the constituent nouns of each complex nominal are assigned. Some example complex nominals plus relation are:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[press] ⇐ (SOURCE) ⇐ [commentaries]
[growth] ⇒ (MEASURE) ⇒ [rate]

[electronic] ⇐ (MEANS) ⇐ [theft]

[campaign] ⇐ (USED_FOR) ⇐ [finances]

Some type of NLP-based machine learning methods can be used to automatically recognise and assign the semantic relations in complex nominals. Using a set of meta-rules some number of relations (about forty-three or so) can be coded so that these can be used to describe the semantic relationships between the constituents.

In conceptual graph matching, the existence of both case-frame relations and complex nominal relations make it possible for the system to detect conceptual similarity even if expressed in different grammatical structures, such as a verb + arguments in a query and a complex nominal in a document, e.g.:

"reduce the debt" = [reduc*] ⇒ (OBJECT) ⇒ [debt]
"debt reduction" = [debt] ⇐ (OBJECT) ⇐[reduc*]

To achieve a better grammatical independence, a further step was necessary in order to match on CRCs produced by verb-based analysis and CRCs produced by complex nominal analysis. This required the determination of the degree of relation-similarity across the two relation sets. There are some x number of relations in case frames and y number of relations in CNs. A relation-similarity table can be constructed that assigns a degree of similarity between some pairs across the two grammatically-distinguished sets, and a degree of similarity between pairs within the same set. The relation-similarity table is used in the final CG matching to allow concepts that are linked by a relation in a document which is different from the relation that links the same two concepts in the query arguments, to still be credited some degree of similarity.

Pre-processing for Conceptual Graph Generation

The processing of phrases after the natural language parsing includes some requirements as follows:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

- Elimination of concept and relation nodes corresponding to contentless meta-phrases (e.g. "Relevant documents must satisfy ..." or "in context of ..."). If both the concept nodes in a concept-relation-concept triple belong to a meta-phrase, the CRC is ignored. When only one of them is a meta-phrase concept, the triple is not removed blindly unless the other concept occurs in another triple.

- Handling of negated parts of concepts . The weights are adjusted in such a way that an occurrence of the negated concept in a document will contribute to the negative evidence that the document will be relevant. In effect, the two weights for the concept are switched.

- Automatic assignment of weights to concept and relation nodes. There are several factors that need to be considered: the conventional way of determining the importance of terms using inverse document frequency (IDF) and total frequency; the location of terms occurring in queries; the part of speech information for each term; and special indications (MUST ...) in the query. But in absence of other sophisticated methods, concept weights based on IDF values may be acceptable.

- Merging common concept appearing in different parts of the query. Although it is not safe in general to assume that two concepts sharing the same concept name actually refer to the same concept instantiation and merge them blindly always, but it may be a fair assumption for queries. In fact, it is desirable to merge Conceptual Graph fragments using common concept nodes. This is an important process that eliminates undesirable effects on scoring. Without this, a document containing a concept occurring repeatedly in a query will be ranked unnecessarily high because each occurrence of the concepts would earn an independent contribution to the overall score.

Ambiguity in Nominal Compounds

The ambiguity of nominal compounds poses a serious difficulty in obtaining head-modifier pairs of highest accuracy. In order to cope with this, the pair extractor looks at the distribution statistics of the compound terms to decide whether the association between any two words (nouns and adjectives) in a noun phrase is both syntactically valid and semantically significant. For example, one may accept

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

language+natural and *processing+language* from *natural language processing* as correct, however, *case+trading* would not be a good term when it is extracted from *insider trading case*. On the other hand, it is important to extract *trading+insider* to be able to match documents containing phrases *insider trading sanctions act* or *insider trading activity*. Phrasal terms are extracted in two phases. In the first phase, only unambiguous head-modifier pairs are generated, while all structurally ambiguous noun phrases are passed to the second phase "as is". In the second phase, the distributional statistics gathered in the first phase are used to predict the strength of alternative modifier-modified links within ambiguous phrases. For example, one may have multiple unambiguous occurrences of *insider trading*, while very few of *trading case*. At the same time, there are numerous phrases such as *insider trading case*, *insider trading legislation*, etc., where the pair *insider trading* remains stable while the other elements get changed, and significantly fewer cases where, say, *trading case* is constant and the other words change.

The disambiguation procedure is performed after the first phrase extraction pass in which all unambiguous pairs (noun+noun or noun+adjective) and all ambiguous noun phrases are extracted. Any nominal string consisting of three or more words of which at least two are nouns is deemed structurally ambiguous. The disambiguation procedure proceeds in three steps, as follows:

1. *Assign scores* to each of the candidate pairs x_i+x_j where $i>j$ from the ambiguous noun phrase $x_1 \ldots x_n$. The score assigned to a candidate pair is the sum of the scores for each occurrence of this pair in any compound nominal within the training corpus. For each occurrence, the score is maximum when the words x_i and x_j are the only words in the phrase, i.e., one has the unambiguous nominal x_ix_j, in which case the score will be 1. For longer phrases, for non-adjacent words, and for pairs anchored at words toward the left of the compound, the score decreases proportionately.

2. For each set $X_j = \{x_i+ x_j | \text{ for } i>j\}$ of candidate pairs *rank* alternative pairs by their scores.
3. *Disambiguate* by selecting the top choice from each set such that its score is above an empirically established global threshold, it is significantly higher than the second best choice from the set, and it is not significantly lower than the scores of pairs selected from other sets X_i.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The effectiveness of this algorithm can be measured in terms of recall (the proportion of all valid head+modifier pairs extracted from ambiguous nominals), and precision (the proportion of valid pairs among those extracted). This disambiguation procedure recovers a few additional pairs, all of which are thrown out as unrecoverable. A sample set of ambiguous phrases and extracted head+modifier pairs is shown in fig 186 below.

Ambiguous nominal	Extracted pairs
oil import fee	oil import import fee
croatian wartime cabinet	croatian cabinet wartime cabinet
national environmental watchdog group	national group environmental group watchdog group
current export subsidy program	current program export subsidy subsidy program
gas operating and maintaining expenses	**gas operating operating expenses maintaining expenses

Figure 185:Ambiguous nominals and extracted pairs

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.7.2.2.4 Relation Concept Detector (RCD)

The output of the Complex Nominal Phraser and the Proper Noun Interpreter modules described before provide concept-relation-concept triples directly to the Relation-Concept Detector (RCD) module. In addition, the following RCD handlers are operative.

One of the distinct features of this system is its capability of extracting and using relations in the final representation of documents and queries in their CG representations. This module provides building blocks for the CG representation by generating concept-relation-concept triples based on the domain-independent knowledge-bases that can easily be constructed from machine-readable resources and corpus statistics. In this module, there are several handlers that are activated selectively depending on the input sentence.

5.7.2.2.4.1 Case Frame (CF) Handler

The main function of the CF Handler is to generate concept-relation-concept triples where one of the concepts comes typically from a verb. It identifies a verb in a sentence and connects it to other constituents surrounding the verb. Since the relations included in our representation are originated from the theories of linguistic case roles and are all semantic in nature, this module will consult the knowledge-base of large number of case frames each of which prescribes a pattern involving a verb and the corresponding concept-relation-concept triples.

Given a set of case frames for different senses of 'decline', for example,

```
(decline 1      ((PATIENT  subject ? obligatory)))
(decline 2      ((AGENT subject human obligatory)
                  (PATIENT object ? optional)))
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(decline 3 ((AGENT subject human obligatory)
(ACTIVITY infinitive ? obligatory)
(link infinitive subject AGENT)))

AGENT, PATIENT and ACTIVITY are the relations that connect the verb to other constituents. The second components (e.g. subject) prescribe the syntactic categories of the constituents and the third components (e.g. human) semantic restrictions that the subject and object should satisfy. The last components (e.g. obligatory) indicate whether the constituent must exist in a sentence in order for the particular case frame to be instantiated. The last line of the third case frame instructs allow the CF handler to produce triples containing non-verbal constituents.

The CF Handler selects the best case frame by attempting to instantiate each case frame and determine which one is satisfied most by the sentence at hand. This can be seen as a sense disambiguation process using both syntactic and semantic information. The semantic restriction information contained in the case frames were obtained from LDOCE, and when the sentence is processed, the CF handler also consults LDOCE to get semantic restriction information for individual constituents surrounding the verb in the sentence and compares it with the restrictions in the case frames of the verb as a way to determine which case frame is likely to be the correct one.

With the following sentence fragment,

... the chairman declined to elaborate on the disclosure ...

the CF handler chooses the third case frame and produces

[decline] ⇒ (AGENT) ⇒ [chairman]

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[decline] ⇒ (ACTIVITY) ⇒ [elaborate]

[elaborate] ⇒ (AGENT) ⇒ [chairman]

the input text to the CF handler is first tagged with part-of-speech information and bracketed for constituent boundaries. A post tagger may be used to attach a part-of-speech tag to individual words (e.g. a\dt brown\adj fox\com noun). The constituent boundary bracketer (such as <>) will mark boundaries of grammatical constituents such as infinitives, noun phrases, prepositional phrases, clauses etc..

5.7.2.2.4.2 Nominalized Verb (NV) Handler

The nominalized verb handler's main function is to consult the NV case frames to identify a NV in a sentence and create concept-relation-concept triples based on the rule. At the same time, it converts the NV into its verb form. In this way, one cam allow for a match between a CG fragment generated from a phrase containing verb and another fragment generated from a noun phrase containing the corresponding nominalized verb. For example, the NV handler converts the sentence fragment

... the company's investigation of the incident ...

into

[investigate] ⇒ (AGENT) ⇒ [company]

[investigate] ⇒(PATIENT) ⇒[incident].

This process is much more than a sophisticated way of performing stemming in that this canonicalizes concept-relation-concept triples than just concept nodes. This NV handler will possess its own case frames for the nominalized verbs.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.7.2.2.4.3 Noun Phrase and Prepositional Phrase (PP) Handler

The noun phrases that are not handled by the complex-nominal handler or by the nominalized verb handler are analyzed so that the head noun is connected to the concepts outside the noun phrase (e.g. a verb concept in the CF Handler). In addition, this module identifies individual concepts corresponding to adjectives and other nouns in a compound noun and connects them with CHARACTESITIC, ATTRIBUTE, or LINK relations. LINK is the most generic relation in our system.

Once noun phrases are handled this way, this module handles prepositional phrases by connecting the head noun concept of the noun phrase to the preceding constituent (e.g. a verb or a noun). The preposition attachment problem is a difficult one, and the current implementation takes the simple-minded approach with general relations such as LINK, which can match with many other semantically more specific relations. This can handle substantial number of prepositional phrase cases. More accurate and finer-level processing will be done with more semantically oriented rules that check the semantic restrictions and use more specific relations. The role of this handler will be diminished when we process phrasal verbs as part of the CF handler, for which case frames are available.

5.7.2.2.4.4 Ad-hoc Handler

This module looks for lexical patterns not covered by any of the other special handlers discussed before. Its processing is also driven by its own knowledge of patterns to infer relations between concepts. For example, a sentence fragment

...bought the item for the purpose of satisfying ...

contains a pattern

[VERB] ... *for the* (ADJ) *purpose of* [NP]

in the knowledge-base, and hence it results in a triple

[buy] ⇒ (GOAL) ⇒ [satisfy]

The knowledge base will contain a small number of simple patterns involving BE verbs and some rules for phrasal patterns across phrase boundaries, by which important relations are extracted. The pattern rules

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

specify certain lexical patterns and the order of occurrences of words belonging to certain part-of-speech categories, and the concept-relation-concept triples to be generated. These patterns require a processing capability no more powerful than a finite state automaton.

5.7.2.2.5 Conceptual Graph (CG) generator

After individual RCD modules have generated concept-relation-concept triples for a document, the CG generator merges those to form a set of conceptual graphs, each corresponding to a clause in most cases. Since more than one handler can generate different triples for the same concept pairs (e.g. a prepositional phrase handled by the CF handler and the NP/PP handler) based on independently constructed rules and on independent processes, a form of conflict resolution is necessary. For simplicity, one can simply order the executions of different handlers based on the quality of the rules and the resulting triples so that more reliable handlers have higher precedence.

The concept nodes in the resulting CGs can not only contain general concept names but also some instantiations (referents) of the concepts. Such a concept can be derived either from a proper noun such as a company name or from a sub-ordinate clause. In the latter case, the instantiation is a CG itself to produce a CG like

[country: {US}] ⇐ (SOURCE-OF-INFO) ⇐ [C#: [[pact] ⇐ (PATIENT) ⇐ ...]

As a way to make the concept representations more "conceptual", one can add a module that will add RIT (Roget's International Thesaurus) codes to individual concept nodes so that the label on the nodes is not a word but a position on the hierarchy in the RIT. The lowest level position beyond the lexical items in the RIT hierarchy is called a semi-colon group consisting of several terms within the delimiter of semo-colons, which represents a concept.

The mapping from a word (called target) in text to a position in RIT requires sense of disambiguation, and the approach suggested is to use the words surrounding the target word as the context

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

within which the sense of the target word is determined and one or more RIT codes are selected. Using this method the minimal number of RIT codes, not just the best one should be selected for the target words because it is generally felt that some of the sense distinctions made in RIT are unnecessarily subtle and it is unlikely that any attempts to make such fine distinctions would be successful and hence contribute to information retrieval.

5.7.2.2.6 Conceptual Graph (CG) Matcher

The main function of the CG matcher is to determine the relevance of each document with a query and produce a ranked list of documents as the output of the system. Using the techniques necessary to model plausible inferences with CGs [103], this module computes the degree to which the query CG is covered by the CGs in the document.

While the most obvious strength of the CG approach is its ability to enhance precision by exploiting the structure of the CGs and the semantics of relations in document and query CGs, and by attempting to meet the specific semantic constraints of the queries, it is also attempted to increase the recall by allowing flexibility in node-level matching. Concept labels can be matched partially (e.g. between 'Bill Clinton' and 'Clinton'), and both relation and concept labels can be matched inexactly (e.g. between 'aid' and 'loan' or between 'AGENT' and 'EXPERIENCER'). For both inexact and partial matches, the degree of matching is determined a multiplication factor of less than 1 is added to the score. For inexact matching cases, a relation similarity table can be used that determines the degree of similarity between pairs of relations. The similarity table can reflect an ontology of relations and allows for matching between relations produced by different RCD handlers whose operations in turn are heavily dependent on the domain-independent knowledge-bases.

While checking various strategies for CG matching/scoring, one common question comes up saying how to "normalize" the score assigned to a document based on the current scoring scheme. The scoring algorithm as described above is query-oriented in the sense that the score reflects how far the query

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

CG is covered by the document CG. While this approach is theoretically justifiable, one drawback is that a document containing the entire query CG is not ranked higher than the document that contains fragments of the query CG scattered all over as long as the whole query CG is covered.

With the intuitive notion that the number of matching CG fragments in a document would be inversely proportional to "connectivity", one can write various formula for the normalization factors that are a function of matching CG fragments. As an example, one may consider 12 sentential CGs as a unit (called a paragraph) and use the number of units containing one or more matching CG fragments in the normalization function. Such a factor can be written as:

1.5 ^ (1-x)

where x is the number of text units that contain one or more matching CG fragments. When this is combined with the maximum of the scores assigned to individual "paragraph" as follows:

S * 1.05 ^ (1-x) + 0.4*M

where S is the unnormalized score and M is the maximum "paragraph" score. But a different combination of constants or a different formula may give still better results.

Term weighting issues

Finding a proper term weighting scheme is critical in term-based retrieval since the rank of a document is determined by the weights of the terms it shares with the query. One popular term weighting scheme, known as tf.idf, weights terms proportionately to their inverted document frequency scores and to their in-document frequencies (tf). The in-document frequency factor is usually normalized by the document length, that is, it is more significant for a term to occur 5 times in a short 20-words document, than to occur 10 times in a 1000-words article.

It is suggested to use this tf.idf weighting scheme for all terms alike: single "ordinary-word' terms, proper names, as well as phrasal terms consisting of 2 or more words. Whenever phrases were included in

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the term set of a document, the length of this document was increased accordingly. This had the effect of decreasing the tf factors for 'regular' single word terms.

A standard tf.idf weighting scheme is inappropriate for mixed term sets (ordinary concepts, proper names, phrases) because:

1. It favors terms that occur fairly frequently in a document, which supports only general-type queries (e.g. "all you know about 'star-wars'"). Such queries are not very typical.

2. It attaches low weights to infrequent, highly specific terms, such as names and phrases, whose only occurrences in a document often decide of relevance. Note that such terms cannot be reliably distinguished using their distribution in the database as the sole factor, and therefore syntactic and lexical information is required.

3. It does not address the problem of inter-term dependencies arising when phrasal terms and their component single-word terms are all included in a document representation, i.e., *launch+satellite* and *satellite* are not independent, and it is unclear whether they should be counted as two terms.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6. Appendix A

Package Management System (PMS): A High Level Interface for Source code management:

6.1 Introduction: This system is used for source code management operating on a package level rather than single programs. In one single command one large set of programs can be retrieved or written back from code library and thus it makes our job of source management much easier. A package can be a combination of many programs, header files,shell programs etc. Normally it may mean an executable.

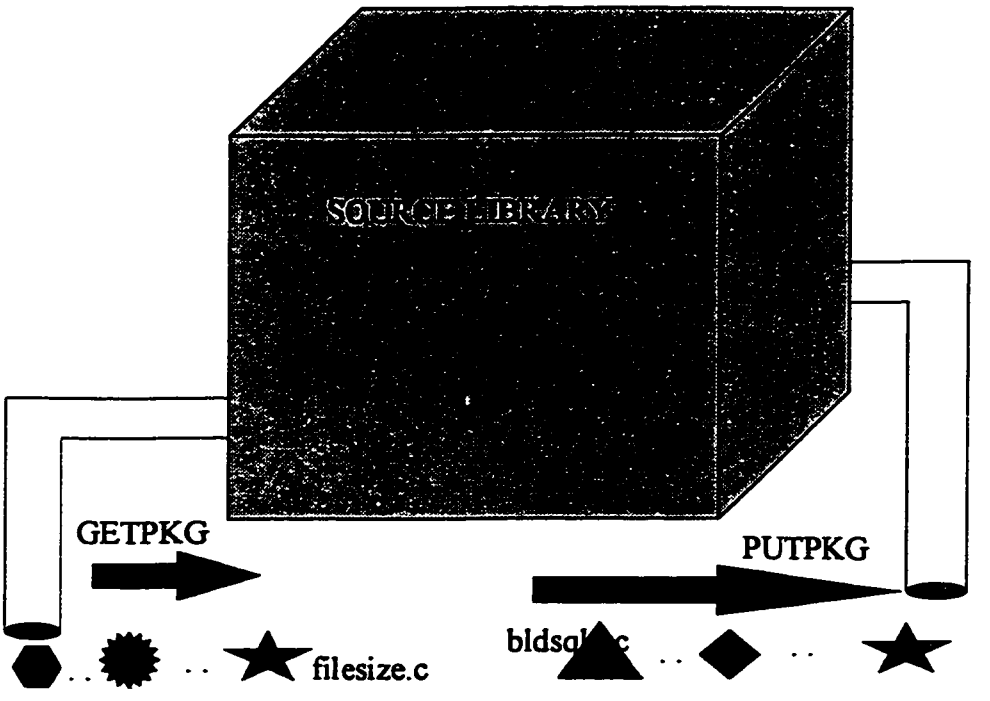


Figure 186:Overview of PMS

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Taking an example, say a group of modules that exist in AVT finally endup in 2 executables (sqlbldr, sqlrptr) and say that we have created the make files to compile them. We can create 2 package configurations called sqlbldr and sqlrptr in PMS where we name all the respective constituent modules (once for sqlbldr and once for sqlrptr). Once such a configuration is defined, to retrieve all the modules in a source package we use the command getpkg . we can put all the modules into source library using one command as putpkg.

6.2 Operations necessary:

1) Define a configuration: This defines the configuration of a package specifying all the component modules contained in the package. User can also define a modified version of a package (dropping a few modules and adding a few modules) if there was a change in the configuration. Thus the system supports multiple versions (some prior configurations and the current configuration) of the same package each containing possibly totally different configurations. The command is:

cfgpkg -{cm} [-v version1,version2,...] Pkgname
ex: cfgpkg -c sqlbldr or cfgpkg -c -v1.10 sqlbldr

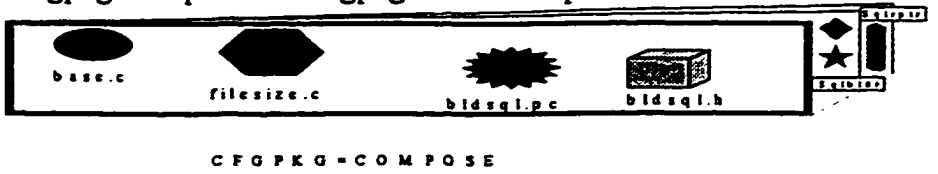


Figure 187:What is cfgpkg?

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2) Retrieve a package: With one command all the modules can be retrieved from source library into any directory. The command is: **getpkg [-r|e|E] [-v version] [-d targetdirectory] [-m module] pkgname**

ex: getpkg sqlbldr or getpkg -d $HOME sqlrptr

3) Check-in a package: With one command the modules that have been modified since their retrieval will be checked into source library. The command is:

putpkg [-v version] [-d targetdir] pkgname

ex: 1) putpkg -d $HOME sqlbldr 2) putpkg -v 2.10 -d. sqlrptr

The default directory is the current directory.

4) Reset a checked-in package: With one command all the modules that have been retrieved from the source library will be unlocked and reset. Handy when a getpkg command was done for edit purpose by mistake. The command is resetpkg [-v version] [-d targetdir] [-m module] pkgname.

6.3 General Principle of Operation:

1) Cfgpkg operation: The user can create a new version (or version 1.0 if none already exists) of a package or modify an existing configuration, then specify all the component modules through a dialog. If he confirms, then all the changes are put into the system. This step is necessary before any other operation can be done on a package.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2) Getpkg operation: The getpkg works on packages that have been configured already. During a getpkg operation, user can specify option -E (means edit all) and that will get every module in edit mode. If the user specifies -r (readonly mode) that will get every module in readonly mode. If the user specifies none (of -E,-r,-e) or -e, it will ask the user if he wants at least one module in edit mode and possibly some in readonly mode. Locks are created inside a registry file with a record of each modulename, sid, the directory, userid, timestamp, editmode(readonly or edit) for each module retrieved. We have a strong lock for retrieval in edit-mode and a weak lock for readonly mode.

How The lock works: The modules that are retrieved for edit, can be modified by the user but the readonly modules will not be checked into library. In case any of the readonly modules were changed at the time of putpkg, a warning will show saying "Warning- readonly module err_hndler has been changed".

At the time of retrieving, the system checks if the module already exists in the target directory or not. (Runs a diff and finds out the difference between the existing file and the retrieved source).If yes, then a message appears on the screen saying " Module xyz already exists in directory /a/b/c- not retrieved again". If it does exist with any difference, system asks for a confirmation to delete them (or else it aborts the run).

If any module of the package has been locked in edit mode, no such module can be locked by another getpkg operation (even for the same user). But the readonly modules can be locked in edit mode by another getpkg operation (otherwise the system will be highly be too serial and thus inefficient resembling to a claim that all the people in the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

world must work serially - only one at a time & thus a better order). But the system can be made to run in that mode also (strict serial mode). In strict serial mode, all modules carry edit-lock. But in soft serial mode, an editable module can be edit-locked once and read-locked many times (carrying 2 types of lock).

In case any read-locked module has been edit-locked and later returned to library, all the packages with readlocked module must retrieve & test with this new module. So at the time of putpkg, we get a message saying "the readlocked module already received an upgrade - So first getpkg -m modulename -v version Pkgname again. In case the readlocked module has not received an upgrade in library - we display a warning "warning! module err_hndler has been earmarked for update by user Joe" and continue.

3) Putpkg operation: The package and its version must already exist in configuration. Putpkg works on three types of modules in a package. Some new modules and some editable updates and some readonly modules.

New modules have no locks. So they are just checked in.

The updateable modules with edit locks will be checked in (with a consistency check run apriori to make sure that the library still has the same SID that was retrieved). Also if no real change had been made on an updateable module (for this the recorded timestamp is compared with the timestamp of last modification), it is not physically checked in. Also if

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the package has any updateable module, it must be checked in from the same directory where it was retrieved into.

The readonly modules are not checked in. But we make sure that they have not been changed since they have been retrieved here. In case a readonly module has been updated in the library, the user must retrieve the module and test everything.

6.4 Concurrency rule: To discuss the valid levels of concurrency, it will be worthwhile to start with two concurrent operations and then we will look at 3 operations.
The timing diagram is displayed below.

a) Two concurrent operations: The various events that can take place are gr (getpkg readonly), ge (getpkg edit), pr (putpkg readonly) and pe (putpkg edit). The ways 2 gets and 2 puts can complete are as either gr or ge to start with (any of 2). Say we caused gr event. Next we can have one of the 2 events (pr or ge). Say we have ge. Now we must have pr and pe in any order. Thus total no of events is 2*(1*(2)+1*1*1) = 2*3=6. These

events are: gr ge pr pe
gr ge pe pr
gr pr ge pe
ge gr pr pe
ge gr pe pr
ge pe gr pr

Now of these, ge gr sequence is not allowed (no readonly access between ge and pe).
So semi-valid sequences are:
gr ge pr pe
gr ge pe pr
gr pr ge pe
ge pe gr pr

Now for sequence gr ge pe pr ,we must make sure that pr = pe.
So valid sequences are: gr ge pr pe, gr pr ge pe, ge pe gr pr and the one to be stopped is

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

gr ge pe pr.

b) 3 concurrent operations: The basic principle is same but we have to make sure that the 3 intervening getpkg requests contain modules so that any common module has the above sequencing scheme satisfied.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

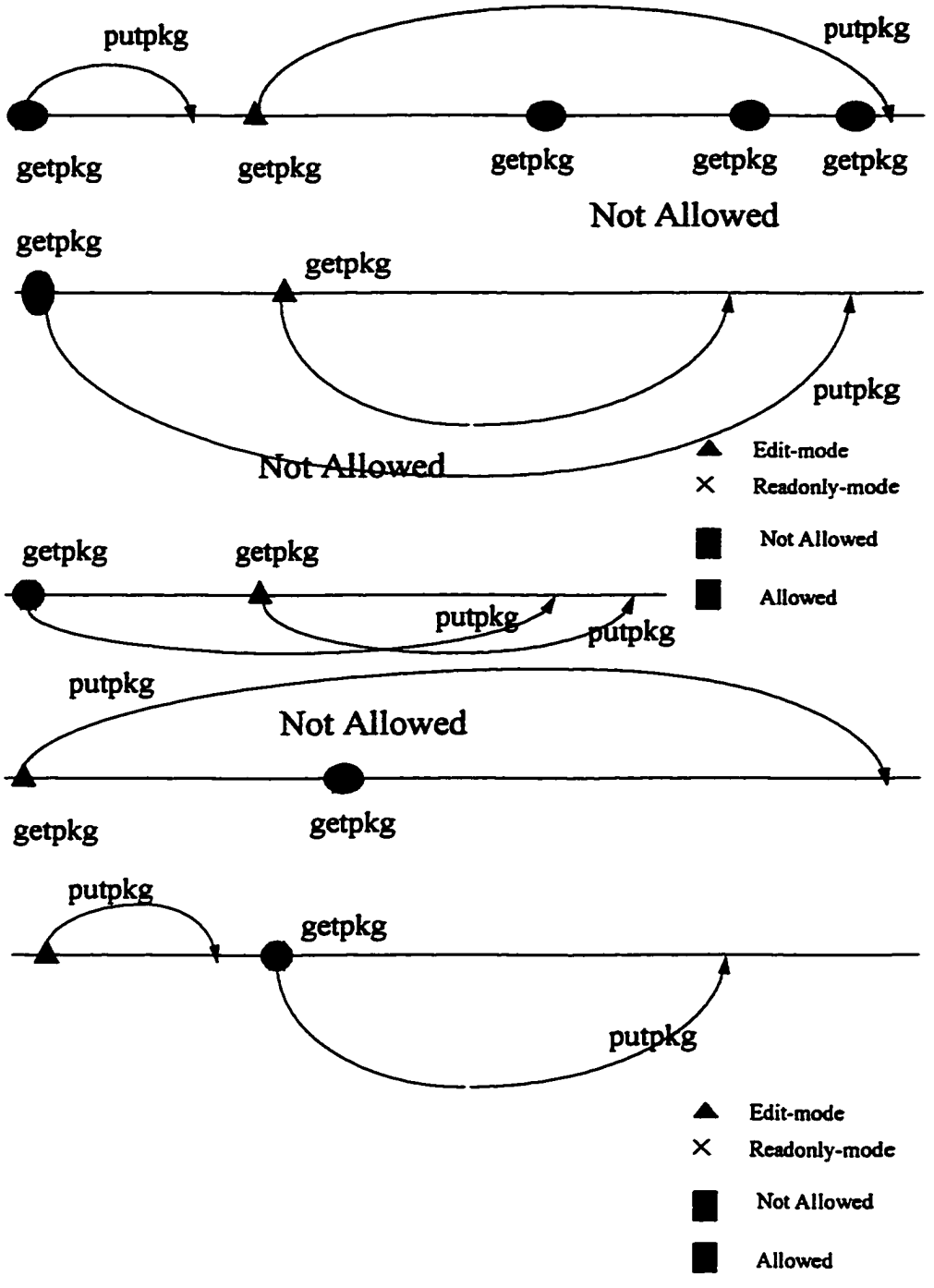


Figure 188:Timing diagrams for valid & invalid lock operations

6.5 Database Layout:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

One database contains all the informations such as package configuration details, package module locks, package module lock history, current environment setup information,consistency check informations etc. In the beginning of any run, a consistency check is conducted to verify if the DB is consistent or not. After that, various informations are loaded into various files. These files are processed individually and at the end of run, all the informations are consolidated into the database again. Such approach makes data portability and user interface very simple. Also the database contains ample chances to write comments making the database highly readable.

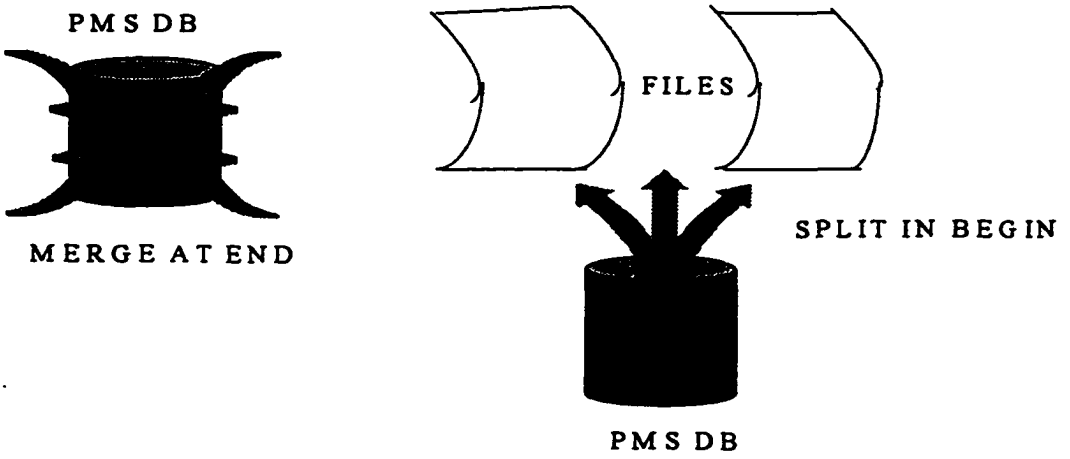


Figure 189:Principles of merge and split

The functionalities & layouts for various files are as follows:

Header file: This file only contains all the header informations used for remembering. The layout is : **Dbname:DbCreator:dateofcreation**

Configuration file: This file contains all the configuration structures. All configuration changes for packages affect this file. The layout is as:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

pkgname version modulename datetimeofcretaion userid sid

Lock file: This file contains all the locks for edit and readonly retrieval of modules (one line for each module that has been locked). The syntax is :

pkgname version modulename sid directory datetime userid editflag

Lock History file: This file contains all the locks that were placed in past and have been released. This syntax is same as the lock file except that a release date (new fld) is added at the end.

Trailer file: This file has all the trailer informations as :

totalCnt(all lines in all files):fileName@cntOfLines:fileName@cntOfLines

6.6 Miscellaneous Rules:

1) In case no version number is specified for a package, the latest existing package number is used or in case of a new package number, it is created as 1.00 for a new package or by adding .01 to the existing latest package version number.

2) For getpkg operation: For all the package level operations, the effective userid is internally set to 'avt'. So in case a target directory is specified, this user ('avt') must have write access on the target directory.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.7 Current limitations:

1) In directory name absolute directory needs to be specified (no relative directory). Also no .. (or parent dir) is allowed.

2) Multiple environments (e.g. multiple tests and production environments) are not supported now, but can be enhanced in future.

Source code in cfgpkgc:

```
static char *VR = "%R%.%L%.%B%.%S% @ %G% %U%";
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>


#define xx() { \
      fprintf(stdout,"%s:%d\n",__FILE__,__LINE__); \
      fflush(stdout); \
}

/* global definitions go here */

/*****************************************************************************
 *
                                    *
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
* <main> - <main this function passes control to Bourne Shell *
*
                                *
*************************************************************************
*
* .Author
*  A.K.Saha - 3/01/97
*
* .Description:
*  This program copies all the environment pointers into a new area,
*  the argv data into a new area and then executes Bourne shell.
*
* .Parameters
*  all the commandline parameters as follows:
*     -cm -v version1,version2,... Pkgname
*
* .Module
*  None
*
* .Exit
*  Under normal circumstances, it should not return.
*  If errored to start the program, exit code is the return code.
*
* .Supplementary:
*          DATE        MODIFIER          DESCRIPTION
*************************************************************************
*    6/5/97          A.K.Saha        Formal template
*
*************************************************************************/
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
main(int argc, char **argv, char **envp){

int i=0,j=0;

char ** newenvp;

char ** newargv;

void *p;

while (envp[i]) i++;


newenvp=(void *) malloc(sizeof(p) *(i+2));

newargv=(void *) malloc(sizeof(p) *(argc+2));

i++;

for (j=0;j<=i;j++) memcpy((void *) ((int)newenvp +  j*(sizeof(p))),

                    (void *) ((int)envp+ j*(sizeof(p))),sizeof(p));

for (i=argc-1; i>=1; i--) newargv[i+1]=strdup(argv[i]);

newargv[argc+1]=(char *) 0;

newargv[0]=strdup("/bin/sh");

newargv[1]=strdup(strcat(strcat("./",argv[0]),".sh"));

execve("/bin/sh",(char *const *) newargv, newenvp);

exit;

}
```

Source of cfgpkgsh:

```
#set -xv
                              *
# ************************************************************************
# *
# * .Author: A.K.Saha
# * .Date:   03/01/97
# * .Description: User can create or modify configuration of a
# *  package. The user can create a new configuration in any of the two
# *  ways:1) By specifying a new packagename that does not already exist
# *       2) By specifying an existing packagename and
# *            a version number that does not exist.
# *  To create a package, user should enter option -c
# *  To modify a package, user should enter option -m
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
# *  User needs to specify the version numbers which he wants to
# *  create or modify.
# *  The systax is : cfgpkg {-cm} [-v version1,version2,..] Pkgname
# * .Parameters
# *    options  : -cm : c is used for creating m for modify
# *                -v ver1,ver2...  : to specify versions
# *    arguments:Pkgname  : Package name.
# * .Modules called: none
# *
# * .Exit
# *  0 - Good
# *  1 - Bad
# *
# * .Supplementary:
# *-----------------------------------------------------------------
# *         DATE        MODIFIER          DESCRIPTION
# *-----------------------------------------------------------------
# *06/05/97         A.K.Saha    Add this header info
# *
# ****************************************************************/

#set  -x
# get the  latest version from SCCS repository
#####################################################################
#####################################################################
#   Function CheckCount
#Check the count of commas in ver list and if count > 1 error & exit
#####################################################################


CheckCount () {

Ver=$1

ret=`echo $Ver| nawk '

 /[, ]/ { print 1; i++;

        }

BEGIN { i=0;j=0}

END {if (i ==0) print 0 }

'`

if [ $ret -eq 1 ]

 then

    echo "Err: Max of 1 version allowed per create- try again"

    echo $USAGE

    exit 1
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
fi

}


#####################################################################
# Function  ChVerNE     # Check that the Version does not exist
# Make sure that the version in $Ver does not preexist in
# the Configuration file.
# filename a.tmp contains the relevant contents of configuration.
# filename b.tmp contains the specified versions.
#####################################################################

ChVerNE () {

 confcount1=`expr $1`

 Pkgname1=$2

 nawkprog='

  BEGIN {result=0; for (i=1; i<=count; i++)

         {getline x <"a.tmp";rr[x]=x;} }

   {if ($1 in rr)

    {print 1; result=1;exit}

    else next

   }

  END {if (result == 0) print 0}

  '

 grep  -e "^$Pkgname1 " "$ConfSection"|cut -d" " -f2  > "a.tmp"

 sort -u -o"a.tmp"  "a.tmp" 2>/dev/null

 if [ ! -s a.tmp ]

   then return 0

 fi

# a.tmp is the Configuration; b.tmp is the version list

ret=`nawk -v count=$confcount1 "$nawkprog" b.tmp`

ret=`expr $ret`

 if [ $ret -eq 1 ]
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
   then echo "Err: verno already exists-cannot create - Please try
again"
   echo $USAGE
   exit 1
 fi
}

#######################################################################
# Function  GuessTheOp
# Tries to guess the operation by checking if verrsion exists in the
# config file or not. set Either modify flag to 1 or create flag
# to 1.
#######################################################################

GuessTheOp () {
 Ver1="$1"
 Pkgname1=$2
 Count=$3
 if [ $modify -eq 1 ]
    then return 0;
 fi
 if [ $create -eq 1 ]
    then return 0;
 fi
 if [ -s "b.tmp" ]
   then wc -l "b.tmp"| read vercount1 x
 else vercount1=`expr 0`
 fi
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
vercount=`expr $vercount`
if [ ! -s "b.tmp" -a -s "a.tmp" ]
 then  modify=1
       return 0
fi
if [ ! -s "b.tmp" -a  ! -s "a.tmp" ]
 then  create=1
       return 0
fi
if [  -s "b.tmp" -a  ! -s "a.tmp" ]
 then  create=1
       return 0
fi
nawkprog='
 BEGIN {result=0; for (i=1; i<=count; i++)
        {getline x <"a.tmp";rr[x]=x; } }
   {if ($1 in rr) next
   else {print 1; result=1;exit}}
 END {if (result == 0) print 0 }
 '

 wc -l "b.tmp"| read vercount x
# a.tmp is the Configuration; b.tmp is the version list
ret=`nawk -v count=$Count "$nawkprog" b.tmp`
ret=`expr $ret`
if [ $ret -eq 0 ]
  then modify=1
fi
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
 if [ $ret -eq 1 -a $vercount -eq 1 ]
   then create=1
 fi
}

#######################################################################
# Function  CheckVer
# Make sure that all the versions  in $Ver do exist in Configuration.
# filename a.tmp contains the relevant contents of configuration.
# filename b.tmp contains the specified versions.
#######################################################################
CheckVer () {
 confcount1=`expr $1`
 Pkgname1=$2
 nawkprog='
  BEGIN {result=0; for (i=1; i<=count; i++)
         {getline x <"a.tmp";rr[x]=x;} }
   {if ($1 in rr) next
    else {print 1; result=1;exit}}
  END {if (result == 0) print 0 }
  '
 grep  -e "^$Pkgname1 " "$ConfSection"|cut -d" " -f2  > "a.tmp"
 sort -u -o"a.tmp"  "a.tmp" 2>/dev/null
 if [ ! -s a.tmp ]
   then
    echo "Err: No Versions exist -try  cfgpkg again"
    echo $USAGE
    exit 1
 fi
# a.tmp is the Configuration; b.tmp is the version list
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
 ret=`nawk -v count=$confcount1 "$nawkprog" b.tmp`
 ret=`expr $ret`
 if [ $ret -eq 1 ]
   then echo "Err: verno not found - Please try again"
   echo $USAGE
   exit 1
 fi
}

#########################################################################
#   Function  ModifyAll
# Go thru all the versions specified on the command line & change
# them all.
#########################################################################
ModifyAll () {
 Ver1=$1
 Pkgname1=$2
 ret=`echo $Ver1| nawk '
 /[, ]/ {
         do
          {
           ++count1; j=sub("[, ]","",$0);
          }
         while (j>0)
        }
BEGIN { count1=0;j=0}
END {if (count1>0) --count1;print count1 }
'`
ret=`expr $ret + 1`
j=`expr 1`
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
while [ $j -le $ret ]

do

# print $Ver1

Ver2=`echo $Ver1|cut -d"," -f$j`

 Modify $Ver2 $Pkgname1

 j=`expr $j + 1`

done


}

#############################################################################
#   Function  UpdatePrior
#This function is called from Create a new config.
# Before we physically add a new configuration,we go back & check if
# any prior version exists there. In that case,
# the latest SIDNO from SCCS is added at the end of each component
# source item for the latest version.
#############################################################################

UpdatePrior () {

#First check if any prior version exists there or not

Pkgname2=$1

#echo "UpdatePrior"

LastVer=`egrep  -e "^$Pkgname2 " "$ConfSection"| sort -r +1 -2

2>/dev/null| \

cut -d" " -f2 |head -1 2>/dev/null`

rm -f x.tmp

egrep -e "^${Pkgname2}[ ]+${LastVer} " "$ConfSection"|cut -d" " -f3

>e.tmp

rm -f x.tmp f.tmp y.tmp 2>/dev/null

chmod 777 "$ConfSection"

rm -f Pkg.new

echo "" >"f.tmp"

cat "$ConfSection" f.tmp > g.tmp
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
mv -f g.tmp "$ConfSection"
chmod 777 "$ConfSection"
rm -f f.tmp
nawk -v pkg=$Pkgname2 -v ver=$LastVer -v confsection=$ConfSection '
   BEGIN { count = 0 ;change = 0;}
   {
    count1 = count + 1
    if (count1 == 1)
       {
       system ("sccs get -g " $1 ">x1.tmp")
       getline  sid <"x1.tmp"
       rm -f "x1.tmp"
       }
    else if (count1 == 2)
       {
       system ("sccs get -g " $1 ">x2.tmp")
       getline  sid <"x2.tmp"
       rm -f "x2.tmp"
       }
    else if (count1 == 3)
       {
       system ("sccs get -g " $1 ">x3.tmp")
       getline  sid <"x3.tmp"
       rm -f "x3.tmp"
       }
    else if (count1 == 4)
       {
       system ("sccs get -g " $1 ">x4.tmp")
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    getline  sid <"x4.tmp"
    rm -f "x4.tmp"
    }
else if (count1 == 5)
    {
    system ("sccs get -g " $1 ">x5.tmp")
    getline  sid <"x5.tmp"
    rm -f "x5.tmp"
    }
else if (count1 == 6)
    {
    system ("sccs get -g " $1 ">x6.tmp")
    getline  sid <"x6.tmp"
    rm -f "x6.tmp"
    }
else if (count1 == 7)
    {
    system ("sccs get -g " $1 ">x7.tmp")
    getline  sid <"x7.tmp"
    rm -f "x7.tmp"
    }
else if (count1 == 8)
    {
    system ("sccs get -g " $1 ">x8.tmp")
    getline  sid <"x8.tmp"
    rm -f "x8.tmp"
    }
else if (count1 == 9)
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    {
    system ("sccs get -g " $1 ">x9.tmp")
    getline  sid <"x9.tmp"
    rm -f "x9.tmp"
    }
else if (count1 == 10)
    {
    system ("sccs get -g " $1 ">x10.tmp")
    getline  sid <"x10.tmp"
    rm -f "x10.tmp"
    }
else if (count1 == 11)
    {
    system ("sccs get -g " $1 ">x11.tmp")
    getline  sid <"x11.tmp"
    rm -f "x11.tmp"
    }
else if (count1 == 12)
    {
    system ("sccs get -g " $1 ">x12.tmp")
    getline  sid <"x12.tmp"
    rm -f "x12.tmp"
    }
else if (count1 == 13)
    {
    system ("sccs get -g " $1 ">x13.tmp")
    getline  sid <"x13.tmp"
    rm -f "x13.tmp"
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    }
else if (count1 == 14)
    {
    system ("sccs get -g " $1 ">x14.tmp")
    getline  sid <"x14.tmp"
    rm -f "x14.tmp"
    }
else if (count1 == 15)
    {
    system ("sccs get -g " $1 ">x15.tmp")
    getline  sid <"x15.tmp"
    rm -f "x15.tmp"
    }
else if (count1 == 16)
    {
    system ("sccs get -g " $1 ">x16.tmp")
    getline  sid <"x16.tmp"
    rm -f "x16.tmp"
    }
else if (count1 == 17)
    {
    system ("sccs get -g " $1 ">x17.tmp")
    getline  sid <"x17.tmp"
    rm -f "x17.tmp"
    }
else if (count1 == 18)
    {
    system ("sccs get -g " $1 ">x18.tmp")
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    getline  sid <"x18.tmp"
    rm -f "x18.tmp"
    }
 else if (count1 == 19)
    {
    system ("sccs get -g " $1 ">x19.tmp")
    getline  sid <"x19.tmp"
    rm -f "x19.tmp"
    }
 else if (count1 == 20)
    {
    system ("sccs get -g " $1 ">x20.tmp")
    getline  sid <"x20.tmp"
    rm -f "x20.tmp"
    }
 else if (count1 == 21)
    {
    system ("sccs get -g " $1 ">x21.tmp")
    getline  sid <"x21.tmp"
    rm -f "x21.tmp"
    }
 else if (count1 == 22)
    {
    system ("sccs get -g " $1 ">x22.tmp")
    getline  sid <"x22.tmp"
    rm -f "x22.tmp"
    }
 else if (count1 == 23)
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    {
    system ("sccs get -g " $1 ">x23.tmp")
    getline  sid <"x23.tmp"
    rm -f "x23.tmp"
    }
 else if (count1 == 24)
    {
    system ("sccs get -g " $1 ">x24.tmp")
    getline  sid <"x24.tmp"
    rm -f "x24.tmp"
    }
 else if (count1 == 25)
    {
    system ("sccs get -g " $1 ">x25.tmp")
    getline  sid <"x25.tmp"
    rm -f "x25.tmp"
    }

 modname[count1] = $1; rev[count1] = sid
 count++
}
END {
 updconfig(pkg,ver,modname,rev,count1)
 if (change == 1) {
   {
    printf ":0\n:g/^$/d\n" >> "a";
    printf ":x\n" >>"a"
   }
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
     system("vi " confsection " <a >/dev/null 2>&1")
      }
    }
    function updconfig(pkg,ver,modname,rev,count1,line1,items,ver1,mod1,
                userid,date,l,found) {
     while (getline line1 <confsection > 0)
       {
        split(line1,items); pkg1=items[1];ver1=items[2];
        mod1=items[3]; userid=items[4];date=items[5];
        if (pkg1 == pkg  && ver1 == ver ) {
           l = 0; found = 0; while (l < count1 && found == 0)
           { l++; if (modname[l] == mod1) found = 1 }
           if (found == 1) {
           printf ":0\n:g/%s %s %s/d\n",pkg1,ver1,mod1 >"a"
           change = 1
printf "\n%s %s %s %s %s %s",pkg,ver,mod1,userid,date,rev[l] >>"Pkg.new"
           }
         }
       }
     }
   ' e.tmp
cat "$ConfSection" Pkg.new >Pkg.tmp 2>/dev/null
mv -f Pkg.tmp "$ConfSection"
chmod 777 "$ConfSection"
RmLstEol "$ConfSection"
}

#########################################################################
#   Function  Modify
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
# Function to modify the components for a given version.
# Get all the modules that needs to be deleted & rewritten.
#####################################################################
Modify () {
echo "Modifying Package=$2 Version=$1"

Ver3=$1

Pkgname1=$2

# listing the module names in file a.tmp

egrep  -e '^$Pkgname1[ ]+$Ver3 ' "$ConfSection"| cut -d" " -f3- > a.tmp

yesno=""

modno=1

change=0

rm -f modlstfile 2>/dev/null

while [ "${yesno}" != "Y" -a "${yesno}" != "y" -a \

        "${yesno}" != "N" -a "${yesno}" != "n" ]

 do

   printf "%s" "Want to Enter a module?Y/N: "

   read yesno


 done

# Outer while do loop to receive all the module names
while [ "${yesno}" = "Y" -o "${yesno}" = "y" ]

  do

          printf "%s" "Enter module name: "

            read module

            while [ "${module}" = "" ]

            do

                  printf "%s" "Enter module name: "

                  read module

            done
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
            egrep  '^$module' a.tmp
            date=`date +%m/%d/%Y:%H:%M:%S`
            userid=`id|cut -d"(" -f2-2 |cut -d")" -f1-1`
            echo $module $date $userid >>modlstfile
            change=1
            yesno=""
            while [ "${yesno}" != "Y" -a "${yesno}" != "y" -a \
                      "${yesno}" != "N" -a "${yesno}" != "n" ]
            do
                  printf "%s" "Want to Enter a module?Y/N: "
                  read yesno
            done
            if [ "${yesno}" = "Y" -o "${yesno}" = "y" ]
            then    modno=`expr $modno + 1`
            fi
  done
mod=1
if [ $change -eq 0 ]
   then return;
fi
nawk  '
    BEGIN{ printf "\n%s\n", "****************************"}
    { print $1,$2,$3}
    END{ printf "\n%s\n", "****************************"}
' modlstfile
confirm=""
while [ "${confirm}" != "Y" -a "${confirm}" != "y" -a \
     "${confirm}" != "N" -a "${confirm}" != "n" ]
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
do
       printf "%s" "Confirm the ModificationsY/N: "
       read confirm
done
# build the editor commands to delete all the existing modnames
if [ "${confirm}" = "N" -o "${confirm}" = "n" ]
   then return 0
fi
cat <<EOF >a
:0
:g/$Pkgname1 $Ver3 /d
:x
EOF
# Do the delete,add operations on a copy of the configuration file
rm -f tmp.conf
dd if="$ConfSection" of=tmp.conf >/dev/null 2>&1
chmod 777 tmp.conf
vi tmp.conf <a >/dev/null 2>&1
rm -f new.conf
count=`nawk   '
    BEGIN { count = 0;}
    END { print count}
    {
     count++
    }
   ' tmp.conf`
nawk -v count=$count  '
     BEGIN { i=0}
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      {
       ++i;
       if (i == count) {
          printf "%s", substr($0,1,length()) >>"new.conf";
         }
       else
          print $0 >>"new.conf"
      }
  ' tmp.conf
dd if=new.conf of=tmp.conf>/dev/null 2>&1
mod=1
      nawk -v count=$modno -v Pkgname=$Pkgname1 -v Ver=$Ver3 '
          {
           printf "\n%s %s %s %s %s",Pkgname,Ver,$1,$2,$3 >>"tmp.conf"
          }
       ' modlstfile
# finally switch the temporary file back to configuration file
      mv -f tmp.conf "$ConfSection"
    chmod 777 "$ConfSection"
}

#########################################################################
#   Function  Create
# Create a new package in the configuration
#########################################################################
Create () {
Ver1=$1
Pkgname1=$2
alreadyExists=$3
yesno=""
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
modno=1
add=0
echo "Creating a new version of package=$Pkgname1 Ver=$Ver1"
rm -f modlstfile 2>/dev/null
while  [ "${yesno}" != "Y" -a "${yesno}" != "y" -a \
          "${yesno}" != "N" -a "${yesno}" != "n" ]
do
      printf "%s" "Want to Enter a Module?Y/N:"
      read yesno

done
#get all the module names modno maintains the count of modules
while [ "${yesno}" = "Y" -o "${yesno}" = "y" ]
do
      module=""
      while [ "${module}" = "" ]
      do
          printf "%s" "Enter Module Name: "
            read module
      done
      date=`date +%m/%d/%Y:%H:%M:%S`
      userid=`id|cut -d"(" -f2-2 |cut -d")" -f1-1`
      echo $module $date $userid >>modlstfile
      add=1
      yesno=""
      while [ "${yesno}" != "Y" -a "${yesno}" != "y" -a \
            "${yesno}" != "N" -a "${yesno}" != "n" ]
      do
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
            printf "%s" "Want to Enter a Module?Y/N: "
            read yesno
      done
      if [ "${yesno}" = "Y" -o "${yesno}" = "y" ]
      then modno=`expr $modno + 1`
      fi
done
#if nothing was aded then return
if [ $add -eq 0 ]
   then return 0
fi
# Display the list of modules on screen and ask for a confirmation
nawk  '
    BEGIN{ printf "\n%s\n", "****************************"}
    { print $1,$2,$3}
    END{ printf "\n%s\n", "****************************"}
' modlstfile
confirm=""
while [ "${confirm}" != "Y" -a "${confirm}" != "y" -a \
        "${confirm}" != "N" -a "${confirm}" != "n" ]
do
   printf "%s" "Confirm the Additions.Y/N: "
   read confirm
done
mod=1
if [ "${confirm}" = "N" -o "${confirm}" = "n" ]
   then exit 1
fi
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
# if create is confirmed and the package already exists,
# the sidno needs to be retrieved from SCCS and stuck to the modules.
# So call UpdatePrior
if [ $alreadyExists -eq 1 ]
   then UpdatePrior $Pkgname
fi
rm -f new.conf
RmLstEol "$ConfSection"
nawk -v count=$count -v Pkgname=$Pkgname1 -v Ver=$Ver1 \
   -v ConfSection=$ConfSection '
  BEGIN {i=0}
  {
   if (count == 0 && i == 0)
      printf "%s %s %s %s %s",Pkgname,Ver,$1,$2,$3 >>ConfSection
   else
      printf "\n%s %s %s %s %s",Pkgname,Ver,$1,$2,$3 >>ConfSection;
   i++
  }
' modlstfile
}

#########################################################################
# Main program begins  here
#########################################################################
USAGE='Syntax:cfgpkg [-{cm}] [-v version1,version2,...] Pkgname '
Ver=""
trap "rm -f a.tmp b.tmp c.tmp d.tmp e.tmp Pkg.new x.tmp 2>/dev/null;
exit" 0 \
 1 2 3
# ini file initializes configuration variables & contains
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
# a few common functions that are loaded into memory.
. pkgini.sh
rm -f *.tmp
SplitConfig $ConfigFile
RmLstEol "$ConfSection"
if [ $# -lt 1 ]
   then echo "Parameters Missing- please try again"
        echo $USAGE;
        ShowPackages
        exit 1
fi
create=`expr  0`
modify=`expr 0`
version=`expr 0`
alreadyExists=`expr 0`
# Parse & Check Options
while getopts cmv: c 2>/dev/null
 do
  case $c in
    c) if [ $create -eq 1 ]
         then echo $USAGE; echo "Wrong options- try again";exit 1
       else
          create=1
       fi;;
    m) if [ $modify -eq 1 ]
         then echo $USAGE; echo "Wrong options- try again";exit 1
       else
          modify=1
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
          fi;;
     v) if [ $version -eq 1 ]
           then echo $USAGE; echo "Wrong options- try again";exit 1
        else
           version=1;Ver=$OPTARG
        fi;;

     \?) echo $USAGE; echo "Wrong options- please try again";exit 1;;
  esac
 done
shift `expr $OPTIND - 1`

#Check syntax
if [ $# -ne 1 ]
   then echo  "Err: Wrong # of Parametrs- please Retry command"
        ShowPackages
        echo  $USAGE
        exit 1
fi
if [ $create -eq 1 -a $modify -eq 1 ]
   then echo  "Err: Use either c or m - not both - please retry "
        echo  $USAGE
        exit 1
fi
Pkgname=$1
echo $Ver | egrep "," >/dev/null
if [ $? -eq 0 ]
   then commaflag=1
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
   else commaflag=0
fi
rm -f a.tmp 2>/dev/null

#Parse supplied version list; Save each version as a line in file
#b.tmp This file will be used later
vercount=1
if [ "${Ver}" != "" ]
 then
  s1=`echo $Ver |cut -d"," -f$vercount`
  while [ "${s1}" != "" ]
   do
    vercount=`expr $vercount + 1`
    echo   $s1 >>  b.tmp
    if [ $commaflag -eq 1 ]; then
       s1=`echo $Ver |cut -d"," -f${vercount}`
    else s1=""
    fi
   done
  vercount=`expr $vercount - 1`
  sort -o"b.tmp" "b.tmp" 2>/dev/null
  wc -l "b.tmp"| read vercount x
 else
  vercount=0
fi
# Check Further Syntax for options and arguments
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
grep  -e "^$Pkgname " "$ConfSection"|cut -d" " -f2 |sort -u -o"a.tmp" -r
\
 2>/dev/null
confcount=`wc -l "a.tmp"|tr -s"[ *]" " "|cut -d" " -f2`
if [ ! -s a.tmp ]
  then if [ $modify -eq 1 ]
      then echo  "Err: No versions  exist - first create a version"
        ShowPackages
        echo  $USAGE
        exit 1
   else create=1
      if [ "${Ver}" = "" ]
         then Ver=1.00
              echo $Ver >b.tmp
      fi
   fi
else
   if [ $create -eq 1 ]
     then if [ "${Ver}" = "" ]
           then Ver=`head -1 a.tmp`; Ver=`echo "$Ver + .01"| bc`
                echo $Ver >b.tmp
          fi
     else
# Smart rtn to guess if user is trying to create or modify
     GuessTheOp       "$Ver"  "$Pkgname" "$confcount"
       if [ "${Ver}" = ""  -a $modify -eq 1 ]; then Ver=`head -1 a.tmp`
          echo $Ver >b.tmp
       fi
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
   fi
fi
#echo  "step 3"
if [ "${Ver}" = "" ]
   then echo  "Err: need to supply a Ver no - please retry"
        echo  $USAGE
        exit 1
fi
# At this point we know what we are going to do- create or modify Pkg
if [ $create -eq 0 -a $modify -eq 0 ]
   then echo  "Err: Must use either c or m option -  please retry "
        echo  $USAGE
        exit 1
fi
if [ $create -eq 1 ]
   then
#  Make sure User specified only one version
     CheckCount $Ver
# Make sure the version does not exist
     ChVerNE $confcount $Pkgname
fi
## Check if all the specified versions(for modify) exist on
configuration
if [ $modify -eq 1  -a "${Ver}" != "" ]
  then
   CheckVer $confcount $Pkgname
fi
egrep -e "^$Pkgname " "$ConfSection">/dev/null
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
if [ $? -eq 0 ]
 then alreadyExists=`expr 1`
fi
#echo  "step4"
# All checks done- Time to process

if [ $modify -eq 1 ]
   then ModifyAll $Ver $Pkgname
   else Create $Ver $Pkgname $alreadyExists
fi
# In case there is any blank line, delete it.
echo ":g/^$/d \n :x" >b.tmp
vi "$ConfSection" <b.tmp >/dev/null 2>/dev/null
sort +0 -1 -o"$ConfSection" "$ConfSection" 2>/dev/null
chmod 777 "$ConfSection"
rm -f a.tmp
rm -f "new.conf" 2>/dev/null
nawk   '
   BEGIN { getline count < "a.tmp";i=0}
   {
       print $0 >>"new.conf"
   }
 ' "$ConfSection"
mv -f new.conf "$ConfSection"
chmod 777 "$ConfSection"

#Finally merge all the sections
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
MergeSections

exit 0


Source code of getpkgc:

static char *VR="%R%.%L%.%B%.%S% @ %G% %U%";


#include <unistd.h>

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#define xx() { \

      fprintf(stdout,"%s:%d\n",__FILE__,__LINE__); \

      fflush(stdout); \

}


/* global definitions go here */

/*****************************************************************************
 *                                                                           *
 * <main> - <main this function passes control to Bourne Shell *
 *                                                                           *
 *****************************************************************************
 *
 * .Author
 *  A.K.Saha - 3/01/97
 *
 * .Description:
 *  This program copies all the environment pointers into a new area,
 *  the argv data into a new area and then executes Bourne shell.
 *
 * .Parameters
 *  all the commandline parameters as follows:
 *     -cm -v version1,version2,... Pkgname
 *
 *
 * .Module
 *  None
 *
 * .Exit
 *  Under normal circumstances, it should not return.
 *  If errored to start the program, exit code is the return code.
 *
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
 * .Supplementary:
 *      DATE        MODIFIER        DESCRIPTION
 ************************************************************************
 *  6/18/97          A.K.Saha        Formal template
 *
 ************************************************************************/

main(int argc, char **argv, char **envp){

int i=0,j=0;

char ** newenvp;

char ** newargv;

void *p;

while (envp[i]) i++;


newenvp=(void *) malloc(sizeof(p) *(i+2));

newargv=(void *) malloc(sizeof(p) *(argc+2));

i++;

for (j=0;j<=i;j++) memcpy((void *) ((int)newenvp +(int) j*(sizeof(p))),

                  (void *) ((int)envp+ j*(sizeof(p))),sizeof(p));

for (i=argc-1; i>=1; i--) newargv[i+1]=strdup(argv[i]);

newargv[argc+1]=(char *) 0;

newargv[0]=strdup("/bin/sh");

newargv[1]=strdup(strcat(strcat("./",argv[0]),".sh"));

execve("/bin/sh",(char *const *) newargv, newenvp);

exit;

}


Source code of getpkgsh:


#!/bin/sh

#/************************************************************************
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
# *                                                                    *
# *                              NOTICE                                *
# *                              ------                                *
# *                                                                    *
# *                        AT&T - PROPRIETARY                          *
# *             Use Pursuant To Company Instructions                   *
# *                                                                    *
# *    THIS SOFTWARE PRODUCT CONTAINS UNPUBLISHED SOURCE CODE          *
# *    OF AT&T WITH ALL RIGHTS RESERVED. IT IS NOT FOR USE OR          *
# *    DISCLOSURE OUTSIDE OF AT&T WITHOUT WRITTEN PERMISSION.          *
# *                                                                    *
# **********************************************************************/

#/**********************************************************************
# *                                                                    *
# * getpkg.sh - Retrieve all modules/files of a package.              *
# *                                                                    *
# **********************************************************************
# *
# * .Author: A.K.Saha
# * .Date:   03/01/97
# * .Description: User can retrieve component modules of a package.
# *  The user can retrieve the modules in readonly mode or edit
# *  mode.
# *       1) By specifying the readonly or edit mode (or the default
# *  is editmode).
# *       2) By specifying a target directory where the modules
# *  should be retrieved into using -d option (or the default is
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
# *  your current directory).
# *       3) By specifying a version number of the package that needs
# *  to be retrieved (or the default is the latest version).
# *  The syntax is : getpkg [-r|e] [-v version1] [-d dir] Pkgname
# * .Parameters
# *    options  : -r|e : e for edit when we retrieve the modules
# *               for edit purpose. r for readonly mode where
# *               we retrieve the modules for read only.
# *
# *               -v version : This is to specify a particular
# *               version of a package name.
# *
# *               -d dir : This is to specify a particular
# *               target directory where the modules should
# *               be brought into.
# *    arguments:Pkgname  : Package name. Must already exist.
# * .Modules called: none
# *
# * .Exit
# *  0 - Good
# *  1 - Bad
# *
# * .Supplementary:
# *-------------------------------------------------------------------
# *     DATE        MODIFIER        DESCRIPTION
# *-------------------------------------------------------------------
# *06/05/97         A.K.Saha    Add this header info
# *
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
# ************************************************************************/
#*****************************************************************
#set -vx

#*********************************************************************#
# Function:CheckVerExists                                             #
# Description: This function checks if the specified version          #
#   exists in the configuration or not. If not Error.                 #
#  parameters: Pkgname,Ver                                            #
#*********************************************************************#

CheckVerExists () {
Pkgname=$1
Ver=$2
egrep -e  "^${Pkgname}[ ]+${Ver}" "a.tmp" >/dev/null
if [ $? -ne 0 ]
 then
  echo "Err:Pkg Ver combination not in configuration.Try again."
  exit 1
fi
}


#*********************************************************************#
# Function:CheckModExists                                             #
# Description: This function checks if the modules already exist      #
# in the target directory or not. If found, It asks the user          #
#  for a confirmation and then wipes out the existing modules.        #
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
# Parameters: Pkgname, Version, target-Directory
#*********************************************************************#

CheckModExists () {
Pkgname=$1
Ver=$2
Dir=$3
if [ $Dirflag -eq 1 ]
   then cp a.tmp ${Dir}
        cd "$Dir"
fi
rm -f b.tmp c.tmp
# Build a list of the component modules for given package,Version
egrep -e  "^${Pkgname}[ ]+${Ver}" "a.tmp"|tr -s "[ *]" " "|cut -d " " -
f3 >"b.tmp"
uniq b.tmp c.tmp
# At this point file c.tmp contains the list of component modules
rm -f b.tmp
# Before switching to target directory- remember the current dir
# so that we can come back there after checking.
sstr=${PWD}
# Now check if names in c.tmp file exist in the target dir or not.
#The existing ones are written on file b.tmp
Nawkprog1='
 {
  system("if [ -s  " $1 " ]; then echo " $1 " >>b.tmp; fi");
 }
'
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
nawk -v Dir=$Dir "$Nawkprog1" c.tmp
#if anything already exists, then ask for confirm
if [ -s b.tmp ]
   then
      echo some of the modules already exist in the directory
      cat b.tmp
      printf "%s:" "Enter Y if you want to overwrite"
      read x
      if [ "$x" = "y" ]
         then x="Y"
      fi
      if [ "$x" != "Y" ]
         then
           echo Error - aborting the run.
           exit 1
      else
         rm -f `cat b.tmp`
         if [ $? -ne 0 ]
           then echo Error - Removing files
                exit 1
         fi
      fi
fi
if [ $Dirflag -eq 1 ]
 then cd "$sstr"
fi

}
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
#*********************************************************************#
# Function:DispList ()                                                #
# Description: Display the list of retrieved modules                  #
# Parameters: dir,PriorVer,Ver                                        #
#*********************************************************************#
#
DispList () {
Dir1="$1"
Ver1="$2"
Nawkprog2='
 BEGIN {printf "***Got the following modules/files -pl check them\n";
       }
 Ver == $2 {
  printf "***%s\n",$3;
 }
'
nawk -v dir="$Dir1" -v Ver="$Ver1"  "$Nawkprog2" "a.tmp"
}



#*********************************************************************#
# Function:WriteReg ()                                                #
# Description: Write into the registration file                       #
# Parameters: dir,PriorVer,Ver                                        #
#*********************************************************************#
#
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
WriteReg () {
Dir1=$1
PriorVer1=$2
Ver1=$3
Date1=`date +%Y%m%d%H%M%S`
userid1=`id|cut -d"(" -f2-2|cut -d")" -f1-1`
rm -f b.tmp
# Get all the modules as complete path -Preparing to do a ls -l
# so that we can cut out the ts part of its last modify date.
Nawkprog='
 Ver == $2 {
  printf "%s/%s\n", dir,$3> "b.tmp";
 }
'
nawk -v dir=$Dir1 -v Ver=$Ver1 "$Nawkprog" "a.tmp"
rm -f c.tmp
ls -l `cat b.tmp`|tr -s"[ *]" " "|cut -d" " -f6-8|tr " " "-">"c.tmp"
Nawkprog2='
 BEGIN {printf "***Got the following modules/files -pl check them\n";
       }
 Ver == $2 {
  printf "***%s\n",$3;
  if (PriorVer == 1)
    {
    getline ts <"c.tmp"
    printf "%s %s %s %s %s %s %s %s\n",$1,$2,$3,$6,dir,date,usr,ts \
          >>RegSection
    }
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
  else
    {
     system ("sccs get -g " $3 ">x.tmp")
     getline  sid < "x.tmp"
     getline ts <"c.tmp"
     printf "%s %s %s %s %s %s %s %s\n",$1,$2,$3,sid,dir,date,
            usr,ts >>RegSection
     system ("rm -f x.tmp")
    }
 }
'
nawk -v dir=$Dir1 -v PriorVer=$PriorVer1 -v date=$Date1 \
     -v usr=$userid1 -v Ver=$Ver1  -v RegSection=$RegSection \
     "$Nawkprog2" "a.tmp"
}


#*********************************************************************#
# Function:CheckLock ()                                               #
# Description:This function checks                                    #
# if any of the modules in the package has been                       #
#   locked by somebody. In that case, prints out an error.            #
#  File RegSection contains alli the Registration info.              #
#  pkgname,version,module,revision-id,directory,date,ts               #
#  parameters: Pkgname,Ver,configfilename.                            #
#*********************************************************************#

CheckLock () {
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
Pkgname2=$1
Ver2=$2
CFilename2=$3
# CFilename stands for config filename
rm -f er.tmp t.tmp
sort -o SSection02 -d +2 -4 "$RegSection" 2>/dev/null
egrep -e "^[^#][^!].*" SSection02 >t.tmp
mv -f t.tmp SSection02
sort -o SSection01 -d +0 -3 +5 "$ConfSection" 2>/dev/null
egrep -e "^[^#][^!].*" SSection01 >t.tmp
mv -f t.tmp SSection01
# Check if any of the modules as in config file already exist on the
# Registration file or not.
Nawkprog='
BEGIN { mod = "~~~~~~~~~~"; rev = "~~~~"; ret = 1;}
 {
  if ($1 == Pkgname && $2 == Ver)
  {
   if (NF > 5)
   {
    while ($3 < mod && $6 < rev && ret)
     {
      do
       ret = (getline line1 <"SSection02")
      while (substr(line1,1,2) == "#!");
      split(line1,items);
      mod = items[3]; rev = items[4];
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      }
     if ($3 == mod && $6 == rev)
        {
         print 1,mod,rev >"er.tmp";
         exit;
        }
    }
   else
    {
     while ($3 < mod && ret)
      {
       do
        ret = (getline line1 <"SSection02")
       while (substr(line1,1,2) == "#!");
       split(line1,items);
       mod = items[3];
      }
     if ($3 == mod )
        {
         print 1,mod >"er.tmp";
         exit;
        }
    }
   }
 }
'
nawk -v Ver=$Ver2 -v Pkgname=$Pkgname2  "$Nawkprog" "SSection01"
if [ -s "er.tmp" ]
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
   then read error modname <er.tmp
    echo  module $modname already locked - try later
        exit 1
fi
rm -f "mod.tmp x.tmp"
#If good so far,now getthe rev-id from SCCS for registration purpose
Nawkprog2='
BEGIN { system ("touch mod.tmp") }
 {
   if (pkg == $1 && ver == $2)
      if (NF > 5)
         printf "%s %s\n",$3,$6 >>"mod.tmp"
      else
         {
          system ("sccs get -g " $3 ">x.tmp")
          getline  sid < "x.tmp"
          printf "%s %s\n",$3,sid >>"mod.tmp"
          system ("rm -f x.tmp")
         }
   else next
 }
'
nawk -v pkg=$Pkgname2 -v ver=$Ver2 "$Nawkprog2" "$ConfSection"
}

#*********************************************************************#
#                                                                     #
# Function:GetPkg ()                                                  #
# Description: This function gets all the modules from the SCCS       #
#              directory. Parameters:Ver,Priorver,Dirflag,Dir         #
#   Notes: Priorver is a flag when 1 indicates that the specified#
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
#    version is a prior version (not the latest current one).    #
# Dirflag when 1 indicates the target directry is specified.     #
#*****************************************************************#

GetPkg () {


Ver2=$1

Priorver2=$2

Dirflag2=$3

Dir2=$4

Readonly2=$5


nawkprog='

Ver == $2 {

    ++i; item=$3; found=1;

    if (Readonly == 0)

     {

    if (Priorver == 1)

       {

        delta="-r"$6;

           print "getvr.sh -e " delta " " item ";" >"new.tmp"

       }

    else

       {

        delta="";

           print "getvr.sh -e " item ";" > "new.tmp";

       }

     }

   else

     {
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    if (Priorver == 1)
       {
        delta="-r"$6;
           print "getvr.sh  " delta " " item ";" >"new.tmp"
       }
    else
       {
        delta="";
           print "getvr.sh  " item ";" > "new.tmp";
       }
     }
   }
BEGIN { found=0}
END { if (found == 0)
      {
      print "Err: failed to locate pkgname version- please try again"
      system("exit");
      }
    }
'
nawk -v Ver=$Ver2 -v Priorver=$Priorver2 -v Dirflag=$Dirflag2 \
     -v Dir=$Dir2 -v Readonly=$Readonly2 "$nawkprog" "a.tmp"
}

#*************************************************************#
# Function:CheckSyntax ()
# Description: This function checks the syntax of the entered
#              input.  Parameters: Dirflag, Verflag,Readonly,
#              Edit,Ver,Dir. First 4 are flags .
#*************************************************************

CheckSyntax () {
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
Dirflag1=$1
Verflag1=$2
Readonly1=$3
Edit1=$4
Ver1="$5"
Dir1="$6"

if [ $Readonly1 -eq 1 -a $Edit1 -eq 1 ]
   then print "Use either option e or o, not both-Please Retry"
        echo $USAGE
        exit 1
fi
if [ $Verflag1 -eq 1 -a -z "$Ver1" ]
   then print "Use a version# with -v option-Please Retry"
        echo $USAGE
        exit 1
fi
# Now process
if [ $Dirflag1 -eq 1 -a -z "$Dir1" ]
   then print "Use a dir name with -d option-Please retry"
        echo $USAGE
        exit 1
fi
}

#*********************************************************************#
# Function:RDirGetPkg ()
# Description: This function registers the target directory for the
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
# getpkg request.It adds a new section on the configuration file
# called REGFORGETPKG Section (file name RegSection) that has:
#          Pkgname Version module Rev-id directory datetime ts
#Parameters:Dirflag,Verflag,Readonly, editflag,Version, Dir,Pkgname
#*************************************************************

RDirGetPkg () {
Dirflag1=$1
Verflag1=$2
Readonly1=$3
Edit1=$4
Ver1="$5"
Dir1="$6"
Pkgname1="$7"
#Check if package has been gotten by someone or not
egrep -e  "^${Pkgname1}[ ]+${Ver1}" "$RegSection" >c.tmp
if [ $? -eq 0 ]
 then
 echo "Err:Pkg already issued to other user. See the following info:"
 cat "c.tmp"
 exit 1
fi
CFilename="$ConfSection"
#Check if any of the component modules have been checked out or not
CheckLock  $Pkgname1 $Ver1 $CFilename
#echo $Pkgname $Ver
}
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
#*********************************************************************
#Main program follows here
#*********************************************************************

#To start with,Setup the exit activities &initialize globl variables.

#trap "rm -f a.tmp b.tmp c.tmp e.tmp er.tmp f.tmp mod.tmp new.tmp

Section* SSection*;exit" 0 1 2 3

USAGE='getpkg [-r|e] [-v version] [-d TargetDirName] pkgname'


# Initialize configuration variables and load common functions

. pkgini.sh

mv -f *.tmp /tmp 2>/dev/null

#Break the Config DB into files

SplitConfig "$ConfigFile"


Ver=""

Dir=""

Readonly=`expr 0`

Edit=`expr 0`

Verflag=`expr 0`

Dirflag=`expr 0`

if [ $# -lt 1 ]

  then echo $USAGE;ShowPackages; exit 1;

fi


# parse & check all the  command line options

while getopts rev:d: c

 do

  case $c in
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    r) if [ $Readonly -eq 1 ]
          then echo $USAGE;echo "Wrong Options- try again";exit 1
       else
          Readonly=`expr 1`
       fi;;

    e) if [ $Edit -eq 1 ]
          then echo $USAGE;echo "Wrong Options- try again";exit 1
       else
          Edit=`expr 1`
       fi;;

    v) if [ $Verflag -eq 1 ]
          then echo $USAGE;echo "Wrong Options- try again";exit 1
       else
          Ver=$OPTARG;Verflag=`expr 1`
       fi;;
    d) if [ $Dirflag -eq 1 ]
          then echo $USAGE;echo "Wrong Options- try again";exit 1
       else
          Dir=$OPTARG;Dirflag=`expr 1`
       fi;;
    \?) echo $USAGE; exit 1;;
  esac
 done

shift `expr $OPTIND - 1`
Pkgname=$1
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
# Determine the target directory
if [ $Dirflag -eq 0 ]
    then Dir="${PWD}"
fi
if [ "$Dir" = "." ]
    then Dir="${PWD}"
fi

Dir=`echo "$Dir"|nawk -v pwd="${PWD}" '
{ if (substr($0,1,1) == ".")
      {
       pwd = pwd "/";
       sub("\.",pwd,$0);
       print $0;
      }
   else
      print $0
}
'`
#Look for any syntax errors
CheckSyntax  $Dirflag $Verflag $Readonly $Edit "$Ver" "$Dir"

# make sure the pkgname already exists in config file
grep  -e "^$Pkgname" "$ConfSection">a.tmp
if [ ! -s a.tmp ]
 then
   echo  "Err: invalid Pkgname - use valid pkgname"
   ShowPackages ;
   exit 1
fi
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
sort -o a.tmp -r +1 -1 a.tmp 2>/dev/null
#Make sure the specified version already exists in the config file
# Variable Ver represents the specified version
if [ "${Ver}" != "" ]
   then CheckVerExists "$Pkgname" "$Ver"
fi
# check if Ver is the specified version is the latest version or not.
Latestver=`head -1 a.tmp | cut -f2 -d " "`
if [ "${Latestver}" != "${Ver}" ]
   then Priorver=1
   else Priorver=0
fi

# if no ver was entered,we want the latest.
if [ "${Ver}" = "" ]
  then
    Ver=$Latestver
fi

# Now Check done - Begin Processing
# Check if any of the component modules exist in the target directory
# If so, ask for a confirmation and then wipe it out.

CheckModExists "$Pkgname" "$Ver" "$Dir"
# Action!
#First register the getpkg request- Enter all the info inthe Registry
# File
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
RDirGetPkg  $Dirflag $Verflag $Readonly $Edit "$Ver" "$Dir" "$Pkgname"


#Now bring inthe component modules of the pkg intothe targetdirectory
#from SCCS library.

GetPkg "$Ver" $Priorver $Dirflag "$Dir" $Readonly

if [ $Dirflag -eq 1 ]

   then  sstr=${PWD}

         cp new.tmp "$Dir"

         cd "$Dir"

fi

cat new.tmp|sh

if [ $? -ne 0 ]

   then echo "Error found - try later "

        exit 1

fi

if [ $Dirflag -eq 1 ]

   then  cd "${sstr}"

fi

if [ $Readonly -ne 1 ]

   then WriteReg "$Dir" $Priorver "$Ver"

else DispList "$Dir" "$Ver"

fi

# Finally merge all the files back into Configuration file

MergeSections;

exit 0
```

Source code for pkginish:

```
#!/bin/sh
#*************************************************************************
# Author: A.K.Saha/Trecom
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
# Description: get a version of a source from the SCCS repository
#*********************************************************************
PROJECTDIR=/icv2/avt; export PROJECTDIR

ConfigFile="Pkg.conf"

HdrSection="Section01"

ConfSection="Section02"

RegSection="Section03"

TrlrSection="Section04"


#*************************************************************
# Function:ShowPackages ()
# Description: This function shows a list of existing packages
#              on the screen. Requires no parameter.
#************************************************************* kk
ShowPackages () {

#Show all the existing packages on the screen

rm -f c.tmp

egrep  -e "^[^S]|($|.[^e])|(.$|..[^c])|(..$|...[^t])|(...$|....[^i])|(..

..$|.....[^o])|(.....$|......[^n])" \

$ConfSection| sort  +0 -1 -u 2>/dev/null |cut -d" " -f1 > c.tmp \

2>/dev/null

egrep -e "[^#][^!]" c.tmp>e.tmp

rm -f c.tmp

echo "************************************"

echo "* CURRENTLY EXISTING PACKAGE NAMES "

echo "************************************"

cat e.tmp

echo "************************************"

}


SplitConfig () {
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
ConfigFile=$1
nawkprog='
BEGIN {count=0; count1=0}
$1 == "Section" {
if (count1 == 0) count1++
else count++
}
END { if (count1 == 0)
       {
        print "Err: No section headings found- please try again";
        system("exit 1");
       }
      else
         print count
    }
'
count=`nawk "$nawkprog" $ConfigFile`
rm -f Section*
csplit -s -f Section $ConfigFile '/Section/' {$count}
# First check if the REGFORGETPKG section already exists or not
if [ ! -s "$RegSection" ]
   then
      echo "Err:No Section REGFORGETPKG in config file-Setit up First"
      exit 1
fi

for x in `ls Section*`
  do
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    sed  '/Section /d' $x> t.tmp
    mv -f "t.tmp" $x
  done

if [ -s "new.conf" ]
 then mv -f new.conf "$ConfSection"
fi
}

#########################################################################
# Function  RmLstEol
# remove the last end of line character from the end of the file
#########################################################################
RmLstEol () {

filename="$1"
count2=`nawk   '
BEGIN { count2 = 0;}
END { print count2}
    {
     count2++
    }
  ' "$ConfSection"`
rm -f new.conf
nawk -v count2=$count2  '
BEGIN { i=0}
{
 ++i;
 if (i == count2) {
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    printf "%s", substr($0,1,length()) >>"new.conf";
   }
 else
    print $0 >>"new.conf"
}
' $filename
rm -f $filename
dd if=new.conf of=$filename >/dev/null 2>&1
chmod 777 $filename
}


MergeSections () {



echo "Section Header" > "h.tmp"
cat "h.tmp" "$HdrSection"> Section001
mv -f Section001 "$HdrSection"
rm -f h.tmp
echo "Section Conf" > "s.tmp"
cat "s.tmp" "$ConfSection" > Section001
mv -f Section001 "$ConfSection"
rm -f s.tmp
echo "Section REGFORGETPKG" > "s.tmp"
cat "s.tmp" "$RegSection" > Section001
mv -f Section001 "$RegSection"
rm -f s.tmp
echo "Section Trailer" > "s.tmp"
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
cat "s.tmp" "$TrlrSection" > Section001
mv -f Section001 "$TrlrSection"
rm -f s.tmp
cat Section0[0-9] >Pkg.conf
rm -f Section00 $HdrSection $ConfSection $RegSection $TrlrSection
echo bye
}

#*********************************************************************#
# Function:CheckVerExists                                             #
# Description: This function checks if the specified version          #
#   exists in the configuration or not. If not Error.                 #
#  parameters: Pkgname,Ver                                            #
#*********************************************************************#

CheckVerExists () {
   Pkgname=$1
   Ver=$2
   egrep -e  "^${Pkgname}[ ]+${Ver}" "a.tmp" >/dev/null
   if [ $? -ne 0 ]
      then
       echo "Err:Pkg Ver combination not in configuration.Try again."
       exit 1
   fi
  }
```

Source code for putpkgc:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
 static char *VR="%R%.%L%.%B%.%S% @ %G% %U%";


#include <unistd.h>

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#define xx() { \

      fprintf(stdout,"%s:%d\n",__FILE__,__LINE__); \

      fflush(stdout); \

}


/* global definitions go here */


/*****************************************************************************
 *                                                                           *
 * <main> - <main this function passes control to Bourne Shell       *
 *                                                                           *
 *****************************************************************************
 *
 * .Author
 *  A.K.Saha - 3/01/97
 *
 * .Description:
 *  This program copies all the environment pointers into a new area,
 *  the argv data into a new area and then executes Bourne shell.
 *
 * .Parameters
 *  all the commandline parameters as follows:
 *
 *
 *
 * .Module
 *  None
 *
 * .Exit
 *  Under normal circumstances, it should not return.
 *  If errored to start the program, exit code is the return code.
 *
 * .Supplementary:
 *      DATE        MODIFIER         DESCRIPTION
 *****************************************************************************
 *  6/18/97          A.K.Saha         Formal template
 *
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
 **************************************************************************/
main (int argc, char **argv, char **envp) {
int i=0,j=0;
char ** newenvp;
char ** newargv;
void *p;

while (envp[i]) i++;
newenvp=(void *) malloc(sizeof(p) *(i+2));
newargv=(void *) malloc(sizeof(p) *(argc+2));
i++;
for (j=0;j<=i;j++) memcpy((void *) ((int)newenvp +(int) j*(sizeof(p))),
                  (void *) ((int)envp+ j*(sizeof(p))),sizeof(p));
for (i=argc-1; i>=1; i--) newargv[i+1]=strdup(argv[i]);
newargv[argc+1]=(char *) 0;
newargv[0]=strdup("/bin/sh");
newargv[1]=strdup(strcat(strcat("./",argv[0]),".sh"));
execve("/bin/sh",(char *const *) newargv, newenvp);
exit;
}
```

Source code for putpkg1:

```
#!/bin/sh
#************************************************************
# Author: A. K. Saha /Trecom  This
# Description: This shell is for writing a package into SCCS
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
#           repository.
#*****************************************************************
#set -vx


#*****************************************************************
# Function:ShowPackages ()
# Description: This function shows a list of existing packages
#           on the screen. Requires no parameter.
#*****************************************************************


ShowPackages () {
#Show all the existing packages on the screen
SplitConfig $1
Pkgname=""
egrep  -e "^[^S][^e][^c][^t][^i][^o][^n]" Section01| sort  +0 -1 2>/dev/null |cut -d" " -f1 >
e.tmp \
 2>/dev/null
uniq e.tmp d.tmp
mv -f d.tmp e.tmp
echo "************************************"
echo "* CURRENTLY EXISTING PACKAGENAMES  "
echo "************************************"
cat e.tmp
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
echo "************************************"
}

#*****************************************************************
# Function:SplitConfig ()                              *
# Description:This function counts no of sections in Pkg.conf *
#          file and then splits Pkg.conf file into so many *
#          files named as Section01,section01,etc.       *
# Assumptions:                                  *
#   Section00 - Nothing                          *
#   Section01 - Configuration list                  *
#   Section02 - Registrations for GetPkg request          *
#   Section03 - others                           *
#*****************************************************************

SplitConfig () {

ConfigFile=$1
nawkprog='
BEGIN {count=0; count1=0}
$1 == "Section" {
  if (count1 == 0) count1++
  else count++
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    }
 END { if (count1 == 0)
        {
      print "Err: No section headings found- please try again";
      system("exit");
        }
      else
       print count >"f.tmp"
    }
'
nawk "$nawkprog" $ConfigFile
read count <"f.tmp"
rm -f Section*
csplit -s -f Section $ConfigFile '/Section/' {$count}
# First check if the DIRFORGETPKG section already exists or not
if [ ! -s "Section02" ]
  then
   echo "Err:No Section DIRFORGETPKG in config file-Setit up First"
   exit 1
fi

for x in `ls Section*`
do
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
 sed  '/Section /d' $x> t.tmp
 mv -f "t.tmp" $x
done
}

#*************************************************************************#
# Function:CheckVerExists                                                 #
# Description: This function checks if the specified version              #
#   exists in the configuration or not.                                   #
#  parameters: Pkgname,Ver                                                #
#*************************************************************************#

CheckVerExists () {
Pkgname=$1
Ver=$2
sstr="^$Pkgname"
sstr=${sstr}"["
sstr=${sstr}" ]+"
sstr=${sstr}"$Ver"
echo "checkVerexists" $sstr
egrep -e  "$sstr" "a.tmp" >/dev/null
if [ $? -ne 0 ]
 then
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
 echo "Err:Pkg Ver combination not found. Try another comb."
 exit 1
fi
}


#*************************************************************************#
# Function:WriteReg ()                                  #
# Description: Write into registration file                 #
# Parameters: dir,PriorVer                              #
#*************************************************************************#

WriteReg () {
Dir=$1
PriorVer=$2
Ver=$3
Date=`date +%m/%d/%Y`
userid=`id|cut -d"(" -f4-4|cut -d")" -f1-1`
Nawkprog='
 Ver == $2 {
 if (PriorVer == 1)

   printf "%s %s %s %s %s %s %s\n",$1,$2,$3,$6,dir,date,usr \
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        >>"Section02"
  else
    {
     system ("sccs get -g " $3 ">x.tmp")
     getline  sid < "x.tmp"
     printf "%s %s %s %s %s %s %s\n",$1,$2,$3,sid,dir,date,
          usr >>"Section02"
     system ("rm -f x.tmp")
    }
 }
'
nawk -v dir=$Dir -v PriorVer=$PriorVer -v date=$Date -v usr=$userid \
          -v Ver=$Ver  "$Nawkprog" "a.tmp"
}

#*****************************************************************#
#                                                 #
# Function:GetPkg ()                              #
# Description: This function gets all the modules from the SCCS  #
#           directory. Parameters:                #
#           Ver,Priorver,Dirflag,Dir              #
#  Notes: Priorver is a flag when 1 indicates that the specified#
#   version is a prior version (not the latest current one).    #
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
# Dirflag when 1 indicates the target directry is specified.    #
#*****************************************************************#

GetPkg () {

Ver=$1
Priorver=$2
Dirflag=$3
Dir=$4

nawkprog='
Ver == $2 {
  ++i; item=$3; found=1;
  if (Priorver == 1)
    {
     delta="-r"$6;
     if ($Dirflag == 1)
       print "getvr.sh -e " delta " -d" Dir " " item ";" >"new.tmp"
     else
       print "getvr.sh -e " delta " " item ";" >"new.tmp"
    }
  else
    {
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        delta="";
        if ($Dirflag == 1)
          print "getvr.sh -e " " -d" Dir " " item ";" >"new.tmp"
        else
          print "getvr.sh -e " item ";" > "new.tmp";
       }
    }
BEGIN { found=0}
END { if (found == 0)
      {
      print "Err: failed to locate pkgname version- please try again"
      system("exit");
      }
    }
'
nawk -v Ver=$Ver -v Priorver=$Priorver -v Dirflag=$Dirflag -v Dir=$Dir \
"$nawkprog" "a.tmp"
}

#*************************************************************#
# Function:CheckSyntax ()
# Description: This function checks the syntax of the entered
#           input.  Parameters: Dirflag, Verflag,Readonly,
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
#           Edit,Ver,Dir. First 4 are flags .
#*****************************************************************

CheckSyntax () {
Dirflag=$1
Verflag=$2
Ver=$3
Dir=$4

if [ $Verflag -eq 1 -a -z "$Ver" ]
  then print "Use a version# with -v option-Please Retry"
       echo $USAGE
       exit 1
fi
# Now process
if [ $Dirflag -eq 1 -a -z "$Dir" ]
  then print "Use a dir name with -d option-Please retry"
       echo $USAGE
       exit 1
fi
}
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
#*****************************************************************#
# Function:MkLstFlsChngd ()
# Description: This function makes a list of files that have been
#       changed since the package was gotten using getpkg.
#       file called DIRFORGETPKG Section (file name Section02)that
#       has:
#       Pkgname Version module Rev id directory date
#*****************************************************************


MkLstFlsChngd () {


Pkgname=$1
Ver=$2
Dir=$3


sort -o c.tmp  +2  c.tmp
nawkprog='
BEGIN {count=0;count2=0}
 {
  if (NF == 7)
  {
   dat1=substr($5,1,8) + 0; dat2=substr($5,9,6) + 0;
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
 system("putpkg2 " $4 "/" $3 ">>b.tmp");
 }
 else
 {
 dat1=substr($6,1,8) + 0; dat2=substr($6,9,6) + 0;
 system("putpkg2 " $5 "/" $3 ">>b.tmp");
 }
 getline ts <"b.tmp"
 ts1=substr(ts,1,8) + 0; ts2=substr(ts,9,6) + 0;
 if ( (ts1 > dat1) || ((ts1 == dat1) && (ts2 > dat2)))
   {
    printf "%s\n", $0 >>"d.tmp"
    count++
   }
 else
   {
    printf "%s\n", $0 >> "f.tmp";
    count2++;
   }
}
END { printf "%d", count >"e.tmp"
    printf "%d",count2 >"ff.tmp"
  }
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
'

rm -f d.tmp e.tmp  b.tmp f.tmp ff.tmp

nawk "$nawkprog" c.tmp

read count <e.tmp

if [ $count -eq 0 ]

  then echo "no real files to write back "

fi

}


#*********************************************************************#

# Function:PutPkg ()

# Description: This function looks up module after module in file

#       d.tmp and puts them into SCCS library. If everything goes ok

#       then each module name is deleted from the registry file.

# Parameters: none

#*********************************************************


PutPkg () {


Priorver=$1

Dir1=${PWD}

nawkprog='
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
BEGIN {Priorver=Priorver+0}
 {
 if (Priorver == 0)
   {
   if (NF == 7)
    {
     printf "putting file %s into SCCS \n", $3;
     system("putvr.sh " $4 "/" $3 ">b.tmp")
    }
   else
    {
     printf "putting file %s Ver=%s into SCCS \n", $3, $4;
     system("putvr.sh " $5 "/" $3 ">b.tmp")

    }
  }
else
  {
   if (NF == 7)
    {
     printf "putting file %s latest into SCCS \n", $3;
     system("putvr.sh  "   $3);
     system("sccs get -g " $3  " >x.tmp");
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        getline sid <"x.tmp"
       }
      else
       {
       printf "putting file %s Ver=%s into SCCS \n", $3,$4;
       system("putvr.sh -r" $4 " "  $3);
       system("sccs get -g " $3  " >x.tmp");
       getline sid <"x.tmp"
       printf "/%s/{\n/%s/{\n/%s/{\ns/%s/%s/\n}\n}\n}\n",
            $1, $2, $3, $7 " " $4, $7 " " sid >>"e.tmp"
    }
  if (NF == 7)
    {
     printf "/%s/{\n/%s/{\n/%s/{\n d\n}\n}\n}\n", $1, $2,
          $3 >>"c.tmp"
     system ("cd " $4);
    }
  else
    {
     printf "/%s/{\n/%s/{\n/%s/{\n/%s/{\n d\n}\n}\n}\n}\n",$1,$2,
          $3, $4 >>"c.tmp"
     system ("cd " $5);
    }
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    }
  system("rm -f " $3);
  system ("cd " Dir1);
 }
'
nawkprog1='
BEGIN {Priorver=Priorver+0}
 {
 if (NF == 7)
    {
     system ("cd " $4);
     system("rm -f " $3);
     printf "/%s/{\n/%s/{\n/%s/{\n d\n}\n}\n}\n", $1, $2,
          $3 >>"c.tmp"
     system ("sccs unedit " $3);
     system("cd " Dir1);
    }
  else
   {
    system ("cd " $5);
    system("rm -f " $3);
    printf "/%s/{\n/%s/{\n/%s/{\n/%s/{\n d\n}\n}\n}\n}\n", $1, $2,
          $3, $4 >>"c.tmp"
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
    system ("sccs unedit " $3);
    system("cd " Dir1);
   }
 }
'
read count <e.tmp
read count1 < ff.tmp
rm -f c.tmp e.tmp x.tmp ff.tmp
if [ $count -gt  0 ]
 then nawk -v Priorver=$Priorver -v Dir1=$Dir1 "$nawkprog" d.tmp
fi
if [ $count1 -gt  0 ]
 then nawk -v Priorver=$Priorver -v Dir1=$Dir1 "$nawkprog1" f.tmp
fi
cd ${Dir1}
if [ -s c.tmp ]
  then  sed -f c.tmp Section02>b.tmp
       mv -f b.tmp Section02
fi
if [ -s e.tmp ]
  then sed -f e.tmp Section01>b.tmp
      mv -f b.tmp Section01
fi
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
echo "ending"
}

#*****************************************************************#
# Function:VerDirPkg ()   Verify Directory Package
# Description: This function verifies the directory for the Pkg.
# Parameteris: Version, Directory, Packagename
#*****************************************************************

VerDirPkg () {

Ver=$1
Dir=$2
Pkgname=$3

userid=`id|cut -d"(" -f4-4|cut -d")" -f1-1`
sstr="^$Pkgname"
sstr=${sstr}"["
sstr=${sstr}" ]+"
sstr=${sstr}"${Ver}[ ]+.*[ ]+.*[ ]+${Dir}[ ]+.*[ ]+${userid}"
egrep -e  "$sstr" Section02 >c.tmp
if [ "$?" != "0" ]
 then
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
 echo "Err:No prior getpkg op done on this Pkg - PutPkg Invalid"
 cat "c.tmp"
 exit 1
fi
#echo $Pkgname $Ver
MkLstFlsChngd "$Pkgname" "$Ver" "$Dir"
}

#Main program part
#Initialization of variables
. init.sh
trap "rm -f a.tmp b.tmp c.tmp e.tmp er.tmp f.tmp mod.tmp new.tmp Section*
SSection*;exit" 0 1 2 3
PROJECTDIR=/icv2/avt; export PROJECTDIR
USAGE='putpkg [-v version] [-d SrcDirName] pkgname'
Ver=""
Dir=""
Verflag=0
Dirflag=0
if [ $# -lt 1 ]
 then echo $USAGE;ShowPackages "Pkg.conf"; exit 1;
fi
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
# parse options
while getopts v:d: c
 do
  case $c in
    v) Ver=$OPTARG;Verflag=1;;
    d) Dir=$OPTARG;Dirflag=1;;
    \?) echo $USAGE; exit 1;;
  esac
 done
shift `expr $OPTIND - 1`
Pkgname=$1
if [ $Dirflag -eq 0 ]
    then Dir=${PWD}
fi
if [ "$Dir" = "." ]
    then Dir=${PWD}
fi
SplitConfig "Pkg.conf"
CheckSyntax  $Dirflag $Verflag "$Ver" "$Dir"
rm -f a.tmp
# Find the package
grep  -e "^$Pkgname" Section01>a.tmp
if [ $? -ne 0 ]
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
 then
   echo  "Err: invalid Pkgname - use valid pkgname"
   ShowPackages "Pkg.conf";
   exit 1
fi
sort -o a.tmp -r +1 -1 a.tmp
# Check if the specified version exists
if [ "${Ver}" != "" ]
   then CheckVerExists "$Pkgname" "$Ver"
fi
# check if Ver is the latest version or not.
Latestver=`head -1 a.tmp | cut -f2 -d " "`
if [ "${Latestver}" != "${Ver}" ]
   then Priorver=1
   else Priorver=0
fi
if [ "${Ver}" = "" ] # if no ver was entered,make it the latest.
  then
    Ver=$Latestver
fi
# Now Process
VerDirPkg   "$Ver" "$Dir" "$Pkgname"
PutPkg    "$Priorver"
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
# Finally merge all the section files back into Pkg.cfg file
echo "Section Conf" > "s.tmp"
cat "s.tmp" Section01 > Section001
mv -f Section001 Section01
rm -f s.tmp
echo "Section DIRFORGETPKG" > "s.tmp"
cat "s.tmp" Section02 > Section002
mv -f Section002 Section02
rm -f s.tmp
cat Section0[0-9] >Pkg.conf
exit 0
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

BIBLIOGRAPHY

1. Paul S. Jacobs, editor, "Text-Based Intelligent Systems. Current Research and Practice in Information Extraction and Retrieval" - Lawrence Erlbaum Associates Publishers, 1992. (MONOGRAPH).

2. C.J.vanRijsbergen, "Information Retrieval", 2nd ed. - Butterworth Publisher, 1979. (BOOK).

3. Gerard Salton, "Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer" - Addison Wesley Publishing Company, 1988. (BOOK).

4. William B Frakes, Ricardo Baeza-Yates , "Information Retrieval: Data Structures & Algorithms" - Prentice Hall, 1992. (BOOK).

5. S.E. Conry, et al., "Multistage Negotiation for Distributed Constraint Satisfaction", IEEE Trans. Systems, Man and Cybernatics, Vol 21, No 6, Nov/Dec 1991. pp 1,462-1,477.

6. R. Davis, R.G. Smith, "Negotiation as a Metaphor for Distributed Problem Solving", Artificial Intelligence, Vol 20, No 1, 1983, pp 63-109.

7. Ronie Lee Bennett, George E. Policello II, "Switching Systems in the 21st Century", IEEE Communications Magazine, Mar 1993, pp 24-28.

8. William J. Barr, Trevor Boyd, Yuji Inoue, "The TINA Initiative", IEEE Communications Magazine, Mar 1993,pp 70-76.

9. N. Natarajan, G.M.Slawsky, "A Framework Architecture for Multimedia Information Networks", IEEE Communications Magazine, Feb 1992, pp 97 - 105.

10. George F. Luger, William A. Stubblefield, "Artificial Intelligence and the Design of Expert Systems", The Benjamin/Cummings Publishing Company, 1989 (BOOK).

11. Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman, "Data Structures And Algorithms", Addison-Wesley Publication, 1985 (BOOK).

12. K. Parsaye & et al. "Intelligent Databases: Object-oriented, Deductive Hypermedia Technologies". New York: Wiley,1989 (BOOK).

13. Jean Tague, Airi Salminen, Charles McClellan, "Complete Formal Model For Information Retrieval Systems", Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research & Development in IR, 1991.

14. E.B. Wendlandt, J.R. Driscoll, "Incorporating a Semantic Analysis into a Document retrieval Strategy", Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research & Development in IR, 1991.

15. Andreas Lux, Paul de Greef, Frank Bomarius, Donald Steiner, "A generic Framework for Human Cooperation", Proceedings of the international Conference on Intelligent and Cooperative Information Systems/IEEE, 1993

16. Terry Gasterland, "Restricting Query Restriction Through User Constraints", Proceedings of the International Conference on Intelligent and Cooperative Information Systems/IEEE, 1993.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

17. Robert Krovetz, "Viewing Morphology as an Inference Process", Proceedings of the Sixteenth Annual International ACM/SIGIR Conference on Research & Development in IR, 1993.

18. Yasushi OGAWA, Ayako BESSHO, Masako HIROSE, "Simple Word Strings as Compound Keywords: An Indexing and Ranking Method for Japanese Texts", Proceedings of the Sixteenth Annual International ACM/SIGIR Conference on Research & Development in IR, 1993.

19. Y. Kodratoff, G. Tecuci, "Learning based on Conceptual Distance", IEEE Trans on Pattern Analysis & Machine Intelligence, Nov 1988, pp 897 - 909.

20. S.V. Ahamed. "Architecture for a Computer System used for Processing knowledge", US Patent No. US05465319, 11/07/1995 from US Patent Office.

21. S.V. Ahamed. Lawrence H, "Architectural consideration in the design of network-based educational system (NBES)" , Proceedings of the Pacific Telecommunication Council, 15th Annual Conf, Jan 1993,
pp 17 - 20.

22. Tibor Gyires, Balakrishnan Muthuswamy, "A Planning Algorithm for Distributed Manufacturing", Proceedings of the International Conference on Intelligent and Cooperative Information Systems/IEEE, 1993, pp 237 - 246.

23. Gerard Salton, Editor, "The SMART RETRIEVAL SYSTEM Experiments in Automatic Document Processing", Prentice Hall,1971 (BOOK).

24. Gerard Salton, M.J. McGill, "Introduction to Modern Information Retrieval", McGraw-Hill Inc, NY, 1983 (BOOK).

25. H. Turtle, W.B. Croft, "Inference Networks for Document Retrieval",Proceedings of the 13th International Conference on Research & Development in Information Retrieval, SIGIR, 1990, pp 1-24.

26. Roger Schank with Peter G. Childers, "The Cognitive Computer" Addison Wesley, 1984 (BOOK).

27. Rebecca Green, "Insights into classification from the cognitive sciences: Ramifications for index languages", Classification Research For Knowledge Representation And Organization, Proceedings of the 5th International Study Conference on Classification Research, 1991, pp 215-222.

28. Rebecca Green, "The expression of syntagmatic relationship in indexing: Are frame-based index languages the answer?", Classification Research For Knowledge Representation And Organization, Proceedings of the 5th International Study Conference on Classification Research, 1991, pp 79-88.

29. Nicholas J. Belkin, W. Bruce Croft, "Information Filtering and information Retrieval: two Sides of the
Same Coin?", Communications Of The ACM, Dec 1992, Vol 35, No.12.

30. Gerard Salton, Harry Wu, "A term weighting model based on utility theory", Information Retrieval Research, Edited by R.N. Oddy, S.E. Robertson, C.J. van Rijsbergen, P.W. Williams, Butterworths Publisher, 1981

31. J. P. Tremblay, P Sorenson. "An Introduction to Data structures with Applications", Second Edition, 1984, McGraw Hill, NY.

32. Karen Sparck Jones, Martin Kay, "Linguistics and Information Science", Academic Press, NY,1973

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

33. Gerard Salton, "Automatic Information Organization and Retrieval", Mcgraw-Hill Book Company. 1968 (book).

34. J. Nagle, "Congestion Control in IP/TCP Internetworks;RFC-896", Internet Requests for Comments. No 896, DDN Network Information Center, January 1984.

35. A. Demers, S. Keshav, S. Shenker [1990], "Analysis and Simulation of a Fair Queuing Algorithm", Internetwork: Research and Experience, Vol 1, No 1, John Wiley & Sons, September 1990. pp 3-26.

36. C. Topolcic , "Experimental Internet stream protocol, Version 2 (ST-II); RFC-1190", Internet Request for Comments. No 1190, Network Information Center, October 1990.

37. L. Zhang, S. E. Deering, D. Estrin, S. Shenker and D. Zappala [1993]. "RSVP: A New Resource ReSerVation Protocol", IEEE Network Magazine, Vol 9, No. 5,September 1993.

38. Derek Lam, Donald C. Cox, Jennifer Widom, "Teletraffic Modeling for Personal Communications Services", IEEE Communications, Feb, 1997.

39. Ramjee Prasad, "Overview of Wireless Personal Communications: Microwave Perspectives", IEEE Communications, April,1997.

40. Yi-Bing Lin, "Modeling Techniques for Large-Scale PCS Networks", IEEE Communications, Feb , 1997.

41. Norihiko Morinaga, Masao Nakagawa, Ryuji Kohno, "New Concepts and Technologies for Achieving Highly Reliable and High-Capacity Multimedia Wireless Communications Systems", IEEE, Communications, Jan,1997.

42. Hiroshi Ishikawa, "Evolving from Narrowband- NTT's Architecture", IEEE Communications, Aug, 1992.

43. Wlater Honcharenko, Jan P. Kruys, David Y. Lee, and Nitin J. Shah, "Broadband Wireless Access", IEEE, Communications, Jan, 1997.

44. N. Griffeth, Y.J. Lin, "Extending Telecommunications Systems: The Feature Interaction Problem", IEEE Computer Magazine, Vol. 26 No 8, Aug, 1993.

45. Syed V. Ahamed, "Intelligent Networks; Encyclopedia of Telecommunications" - 2989, pp. 159 - 174.

46. Syed V Ahamed, Victor B Lawrence, "Intelligent Networks: Architecture And Implications; Encyclopedia Of Physical Science & Technology", Vol. 8 - Academic Press, 1992, pp. 229 - 262.

47. Roger K. Berman, John H. Brewster, "Perspectives on the AIN Architecture", IEEE Communications Magazine, Feb 1992, pp. 27-32.

48. William Stallings, "Data & Computer Communication", 3rd Ed, Macmillan Publishing Company, 1991
(BOOK).

49. M.F. Slana, H.R. Lehman, "Data Communication Using The Telecommunication Network", IEEE Computer Magazine, May, 1981.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

50. Dominique Delisle, Lionel Pelamourgues, "B-ISDN and how it works", IEEE Spectrum Magazine, Aug, 1991.

51. Craig Partridge, "Gigabit Networking" - Addison-Wesley Publishing Company, 1994 (BOOK).

52. Merlo Gerla, Joseph A Bannister, "High-Speed local-area networks", IEEE Spectrum Magazine, Aug. 1991.

53. Tetsuya Miki, "The Potential of Photonic Networks", IEEE Communications Magazine, Dec, 1994.

54. Anthony S. Acampora, "The Scalable Lightwave Network", IEEE Communications Magazine, Dec, 1994.

55. Ronald J. Vetter, "ATM Concepts Architecture & Protocols", CACM, Feb 1995, pp. 30-38.

56. J. B. Kim, P. Wang, "ATM Networks: Goals & Challenges", CACM, Feb, 1995, pp. 39 - 44.

57. James W. Roberts, "Variable-Bit-Rate Traffic Control in B-ISDN", IEEE Communications Magazine, Sept, 1991, pp. 50-56.

58. R.Ballart, Y-C Ching, "SONET. Now it's the standard optical network", IEEE Communications Magazine, Mar,1989, pp. 8-15.

59. George H. Forman, John Zahorjan, "The Challenges of Mobile Computing", IEEE Computer Magazine, Apr 1994, pp. 38-47.

60. C. Kimberly Claffy, Hans-Werner Braun, C. George Polyzos, "Tracking Long-term growth of the NSFNET", CACM, AUG 1994, pp. 34-45.

61. Helen A. Bauer, John J. Kulzer, Edward G. Sable, "Designing Service-Independent capabilities for Intelligent Networks", IEEE Communications Magazine, Dec 1988, pp. 31-41.

62. Peter A. Russo et al, "IN Rollout in the United States", IEEE Communications Magazine, Mar 1993, pp. 56-63.

63. Emilio cancer, Ron McCann, Marc Aboudharam, "IN Rollout in Europe", IEEE Communications Magazine, Mar 1993, pp. 38-47.

64. James J. Garrahan et al, "Intelligent Network Overview", IEEE Communications Magazine, Mar 1993,
pp. 30-36.

65. David A. Pezzuti, "Operations Issues for Advanced Intelligent Networks", IEEE Communications Magazine, Feb 1992, pp. 58-63.

66. Richard J. Smith, Mark Gibbs, "Navigating The Internet", Deluxe Edition, Sams Publishing, 1994 (BOOK).

67. Tom Bowen, Gita Gopal, Gary Herman, Jr. William Mansfield, "A Scale Database Architecture for Network Services", IEEE Communications Magazine, Jan 1991, pp. 52-59.

68. Michael Cain, "Managing Run-Time Interactions Between Call-Processing Features", IEEE Communications Magazine, Feb, 1992, pp. 44-50.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

69. Jose M. Duran, John Visser, "International Standards for Intelligent Networks", IEEE Communications
Magazine, Feb, 1992, pp. 34-42.

70. E. Fletcher Haselton, "Service-Creation Environments for Intelligent Networks", IEEE Communications Mag, Feb, 1992, pp. 78-81.

71. Roger K. Berman, John H. Brewster, "Perspectives on the AIN Architecture", IEEE Communications Magazine, Feb, 1992, pp. 27-32.

72. Scott Higgs, "Windows to the Internet", BYTE Magazine, July, 1995, pp. 142-148.

73. Nicholas Baran, "The Greatest Show on Earth", BYTE Magazine, July, 1995, pp. 69-86.

74. Jorge A. Brana, Anand S. Ramani, Pantelis H. Vapheas, "Handling Switch-detected Application_Level
Errors in AIN",IEEE Communications, Feb, 1992, pp. 52-58.

75. Daniel Stevenson, Nathan Hillery, Greg Byrd, "Secure Communications in ATM Networks", CACM, Feb, 1995, pp. 45-52.

76. J.D. Ullman, "Principles of DATABASE and KNOWLEDGE-BASE SYSTEMS", Vol. 1, Computer Science Press, 1988 (BOOK).

77. Shivi Fotedar, Mario Gerla, Paola Crocetti, Luigi Fratta, "ATM Virtual Private Networks", CACM, Feb, 1995, pp. 101- 109.

78. A. Iwata, N.Mori, C. Ikeda, H. Suzuki, M. Ott, "ATM Connection and Traffic Management Schemes for Multimedia Internetworking", CACM, Feb 1995, pp. 72-89.

79. Hugo Velthuijsen, "Distributed Artificial Intelligence for Runtime Feature-Interaction Resolution", IEEE Computer magazine, Aug, 1993, pp. 48-55.

80. Grady Booch, "Object-Oriented Analysis & Design with Applications", 2nd Ed, The Benjamin Cummings Publishing Company, 1994 (BOOK).

81. Wilf Lalonde, "Discovering Smalltalk", The Benjamin Cummings Publishing Company, 1994 (BOOK).

82. Richard Frost, "Introduction to Knowledge Base Systems", Macmillan Publishing Company, NY, 1986
(BOOK).

83. Frank R. Hickman, Jonathan L. Killin, Lise Land, Tim Mulhall, David Porter and Robert M. Taylor, "Analysis for Knowledge-Based Systems: a practical guide to the KADS methodology", Ellis Horwood
Limited, Halsted Press A division of John Wiley & Sons , 1989 (BOOK).

84. W. J. Clancey, "The epistemology of a rule-based expert system: a framework for explanation", Artificial Intelligence 20 (3), 215-252, 1983.

85. M.P. Georgeff, "Procedural control in production systems", Artificial Intelligence 18, 175-201, 1982.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

86. F. Hayes-Roth, V. R. Lesser. "Focus of attention in the HEARSAY-III system", Proceedings of the Fifth IJCA. Massachusetts Institute Of Technology, Cambridge, Massachusetts.

87. R. C. Schank, "Conceptual Information processing", New York: Elsevier, North Holland. 1975.

88. T. Gaasterland, "Cooperative Answers for Database Queries", Ph.D. Thesis, University of Maryland, Department of Computer Science, College Park, October 1992.

89. J. Minker, A. Gal., "Producing cooperative answers in deductive databases", Saint-Dizier and S. Szpakowics, editors. *Logic and Logic Grammar for Language Processing.* L. S. Horward. Ltd., 1990.

90. S. Jerrold Kaplan, "Appropriate responses to inappropriate questions", In A.K. Joshi, B.L. Webber, I. Sag, editors. *Elements of Discourse Understanding,* chapter 5, pg 127-144. Cambridge University Press, 1981.

91. S. Jerrold Kaplan, "Cooperative responses from a portable natural language database query system'. In M. Brady, editor, *Computational Models of Discourse,* The MIT Press, Cambridge, MA, 1982.

92. H. Paul Grice, "Logic and Conversation" In *Syntax and Semantics 3:Speech Acts.* Academic Press, New York, 1975.

93. Aravind K. Joshi, "Mutual beliefs in question answering systems", In N. Smith, editor, *Mutual belief,* Academic Press, New York, 1982.

94. Aravind K. Joshi, Bonnie L. Webber, and Ralph Weischedel, "Preventing false inferences", In Proceedings of the 10th international conference on computational linguistics, pp 134-138, 1984.

95. Aravind K. Joshi, Bonnie L. Webber and Ralph Weischedel, "Living up to expectations: Computing expert responses." s", In Proceedings of the 4th National Conference on Artificial Intelligence, pp 169-175, 1984.

96. Aravind K. Joshi, Bonnie L. Webber and Ralph Weischedel, "Default reasoning in interaction." In Proceedings of 1984 workshop on Non-monotonic reasoning, Oct 1984.

97. Aravind K. Joshi, Bonnie L. Webber and Ralph Weischedel, "Some aspects of default reasoning in interactive discourse", In R. G. Reilly, editor, *Communication Failure in Dialogue and Discourse,* pp
213-219, Elsevier, 1987.

98. James P. Allen, "Recognizing intentions from natural language utterances", In M. Brady and R. C. Berwick, editors, *Computational Models of Discourse,* pp 107-166. The MIT Press, Cambridge, MA, 1983.

99. Barbara J. Grosz, Candace L. Signer, "Attention, Intention and the structure of discourse", *Computational Linguistics,* 12(3): 175-204, 1986.

100. Barbara J. Grosz, Candance L. Sidner, "Plans for discourse", In P. R. Cohen, J. Morgan, M. E. Pollack , editors, *Intentions in Communication,* Ch 20, pp 417-444, The MIT Press, Cambridge, MA, 1990.

101. van Dijk, "News as discourse", Hillsdale Lawrence Erlbaum Associates, 1988 (Book).

102. Naomi Sager, "Natural Language Information Processing", Addison-Wesley, 1981 (Book).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

103. S. H. Myaeng, Chris Khoo, "On uncertainty handling in plausible reasoning with conceptual graphs",
Proceedings of the 7th Conceptual Graphs Workshop, Las Cruces, NM, 1992.

104. N. Fuhr, C. Buckley, "Optimizing Document indexing and Search Term weighting Based on Probabilistic Models", The *First Text REtrieval Conference*" (TREC-1), pages, 89-100, 1993.

105. N. Fuhr, "Models for retrieval with probabilistic indexing", Information processing and management, 25(1): 55-72, 1989.

106. Mark Zimmerman, "Proximity-correlation for document ranking: The PARA Group's TREC Experiment.", Proceedings of the First Text Retrival Conference (TREC-1)", NIST Special Publication , 1992.

107. N. J. Belkin, C. Cool, W. B. Croft, J.P. Callan (1993, June),"The effect of Multiple Query Representations on Information Retrieval Performance", Proceedings 16th International Conference on R&D in IR (SIGIR 1993), Pittsburgh.

108. L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, "Classification and regression trees", Pacific Grove, CA:Wadsworth & Brooks;1984.

109. R.M. Tong, A. Winkler, P. Gage, "Classification trees for document routing". In: Harman, D. K., ed. The First Text Retrieval Conference (TREC-1)", Gaitherburg, MD:March 1993.

110. J. R. Quinlan, C4.5:Programs for machine learning. San Mateo, CA:Morgan Kaufman; 1992.

111. C. D. Paice, "Information Retrieval and the Computer", London, 1977.

112. J. Pearl, "Probabilistic Reasoning in Intelligent Systems", San Mateo, CA: Morgan Kaufman, 1988.

113. H. R. Turtle, W. B. Croft, "Evaluation of an Inference network-based retrieval model", ACM Transactions on Information Systems, 9(3): 187-222, 1991.

114. J. J. Rocchio, "Relevance feedback in information retrieval", In Gerard Salton, editor, "The SMART Retrieval System - Experiments in Automatic Document Processing", Prentice Hall, NJ, 1971.

115. C. J. VAN RIJSBERGEN, "A new theoretical framework of information retrieval", In: Proceedings of the 1986 International Conference on Research and Development in Information Retrieval (SIGIR, 1986), Pisa, 1986, New York.

116. E. A. FOX et. al., "Combining evidence from multiple searches", In: D. K. Harman, ed, *The First Text Retrieval Conference (TREC-1)*, 1993, Washington D.C.

117. E. Cooper, "Broadband Network Technology", Mountain View, CA: Sytek Press, 1984 (BOOK).

118. C. Dhas, U. Konangi, "X.25: An interface to Public Packet networks", IEEE Communications Magazine, Sept, 1986.

119. Edger W. Dijkstra, "A Note on Two Problems in Connection with Graphs", Numerical Mathematics, Oct 1959.

120. I. Dorros, "Telephone Nets Go Digital", IEEE Spectrum, April , 1983.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

121. T. Ha, "Digital Satellite Communications", New York: Macmillan, 1986 (BOOK).

122. W. Hsieh, I. Gitman, "Routing Strategies in Computer Networks", Computer, June, 1984.

123. A. Tanenbaum, "Computer Networks", Second Edition, Englewood Cliffs, NJ, Prentice-Hall, 1988.

124. F. Kuo, "Protocols and Techniques for Data Communication Networks", Englewood Cliffs, NJ: Prentice Hall, 1981.

125. D. Kostas, "Transition to ISDN-An Overview", IEEE *Communications Magazine*, Jan, 1984.

126. W. Chou, ed. "Computer Communications, Vol I: Principles" ,Englewood-Cliffs, NJ:Prentice-Hall, (BOOK)

127. W. Chou, ed. "Computer Communications, Vol II: System and Applications", Englewood-Cliffs, NJ: Prentice-Hall, 1985.(BOOK)

128. D. Bertsekas, R. Gallager, "Data Networks", Englewood-Cliffs, NJ: Prentice-Hall, 1987.

129. Theodore Zahariadis, Corrado Rosa, Marco Pellegrinato, Anita Bjork Lund, George Stassinopoulos, "Interactive Multimedia Services to Resudential Users", IEEE Communications Magazine, June, 1997.

130. Hario Akimaru, Marion R. Finley, Zhiseng Niu, "Elements of the emerging broadband information highway", IEEE Communications Magazine, June,1997.

131. Bijan Jabbari, "Intelligent Network Concepts in Mobile Communications", IEEE Communications Magazine, Feb, 1997.

132. Sami Tabbane, "Location Management Methods for Third-Generation Mobile Systems", IEEE Communications Magazine, August 1997.

133. Ian F. Akyldiz, Seong-Ho Jeong, "Satellite ATM Networks: A Survey", IEEE Communications Magazine, July 1997.

134. Douglas Niehaus, Abdella Battou, Andrew McFarland, Basil Decina, Henry Dardy, Vinai Sirkay, Bill Edwards, "Performance Benchmarking of Signaling in ATM Networks", IEEE Communications Magazine, August 1997.

135. Gotz Brasche, Bernhard Walke, "Concepts, Services and Protocols of the New GSM Phase 2+ General Packet Radio Service", IEEE Communications Magazine, August 1997.

136. Tadanobu Okada, Hirokazu Ohnisi, Naotaka Morita, "Traffic Control in Asynchronous Transfer Mode", IEEE Communications Magazine, Sept 1997.

137. Kunyan Liu, David W. Petr, Victor S. Frost, Hongbo Zhu, Cameron Braun, William L. Edwards, "Design and Analysis of a Bandwidth Management Framework for ATM-Based Broadband ISDN", IEEE Communications Magazine, May 1997.

138. Hiroshi Saito, "Dynamic Resource Allocation in ATM Networks", IEEE Communications Magazine, May 1997.

139. Manfred Wittig, "Large-Capacity Multimedia Satellite Systems", IEEE Communications Magazine, July 1997.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

140. Jim Gilderson, Jafaar Cherkaoui, "Onboard Switching for ATM via Satellite", IEEE Communications Magazine, July 1997.

141. Ian F. Akyildiz, Keith L. Bernhardt, "ATM Local Area Networks: A Survey of Requirements, Architectures. and Standards", IEEE Communications Magazine, July 1997.

142. James W. Roberts, "Variable-Bit-Rate Traffic Control in B-ISDN", IEEE Communications Magazine, Sept 1991.

143. Karen Sparck-Jones, B. K. Boguraev, "How to drive a database front-end using general semantic information", In Proccedings of the Conference on Applied Natural Language Analysis, Santa Monica, California, 1983.

144. J. Mylopoulos, "An overview of knowledge representation", ACM SIGMOD Rec 11(2), 5-12.

145. D. P. Snyder, "Modal Logic and its Applications", New York, Van Nostrand., 1971.

146. P. S. Sell ,"Expert Systems: A Practical Introduction", London and New York, Macmillan, 1985 (BOOK).

147. R. F. Simmons, D. Chester, "Inferences in quantified semantic networks", In Proceedings of the Fifth IJCA. Massachusetts Institute of Technology, Cambridge, Massachusetts, 1977.

148. D. E. Smith, J. E. Clayton, "A frame-based production system architecture", In Proccedings of the AAAI 80. Stanford University, California, 1980.

149. L. K. Schubert, "Extending the expressive power of semantic networks", Artificial Intelligence 7, 89-124, 1976.

150. N. Wirth, "Algorithms + Data Structures = Programs", Englewood Cliffs, New Jersey:Prentice Hall, 1976 (BOOK).

151. L. A. Zadeh, "Commonsense Knowledge representation based on fuzzy logic", Computer, 16(10), 61-65.

152. L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility", Fuzzy Sets and Systems, Amsterdam: North Holland, 1978 (BOOK).

153. W. Richard Stevens,"Unix Network Programming", Englewood-Cliffs, NJ: Prentice-Hall,1990 (BOOK).

154. Dr. Dimitris N. Chorafas, Heinrich Steinmann, "Intelligent Networks Telecommunications Solutions for the 1990s", CRC Press,1990 (BOOK).

155. Joseph JaJa, "An Introduction to Parallel Algorithms", Addison Wesley, 1992 (BOOK).

156. Anton Meijer, Paul Peeters, "Computer Network Architectures", Computer Science Press, 1982 (BOOK).

157. Roshan Lal Sharma, Paulo J. T. deSousa, Ashok D. Ingle, "Network Systems Mdelling, Analysis and Design", Van Nostrand Reinhold, 1982.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

158. B. W. Stuck, E. Arthurs, "A Computer & Communications Network Performance Analysis Primer", Englewood-Cliffs, NJ: Prentice Hall, 1985.

159. Syed V. Ahamed, Victor B. Lawrence, "Knowledge Processing system employing Confidence levels",
Patent No. US05809493, 09/15/1998 from US Patent Office.

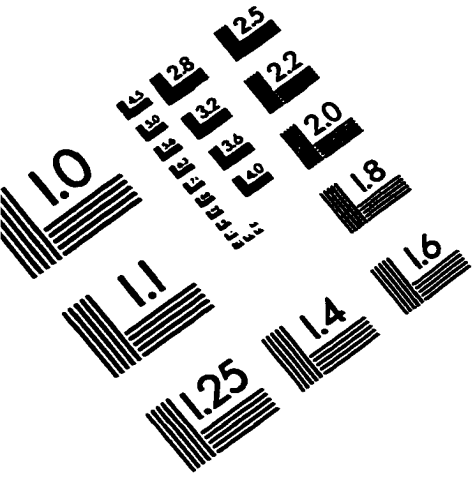
160. Syed V. Ahamed, Victor B. Lawrence, "Personal Computer-based intelligent networks", Patent No. US05715371, 02/03/1998 from US Patent Office.

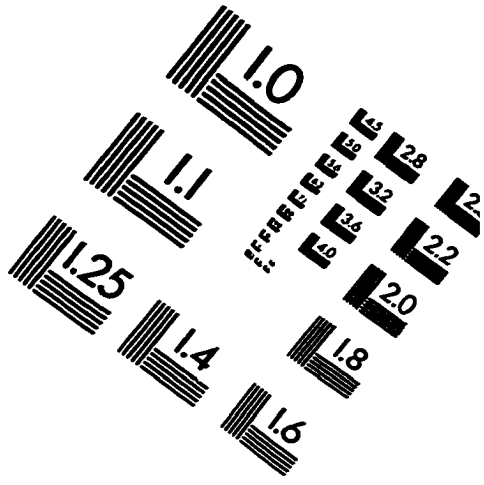
161. Syed V. Ahamed, Victor B. Lawrence, "Network-based intelligent information-sourcing arrangement",
Patent No. US05628011, 05/06/1997 from US Patent Office.

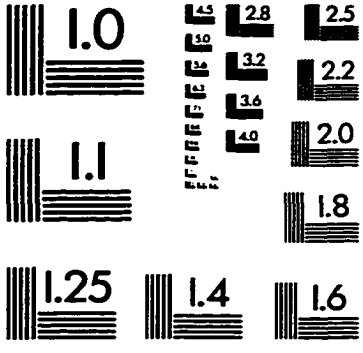
162. Syed V. Ahamed, "Burst Error Correction using Cyclic block Codes", Patent No. US04488302, 12/11/1984 from US Patent Office.

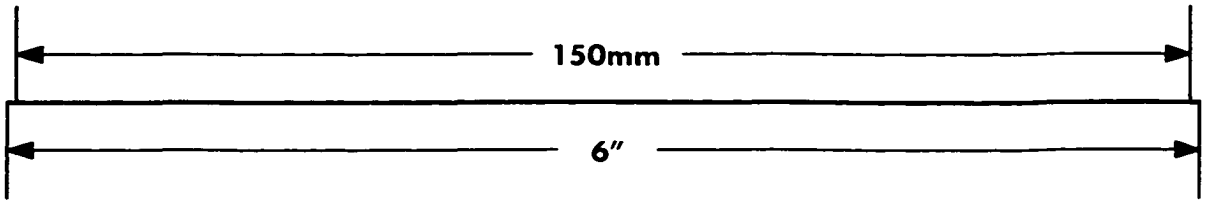
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

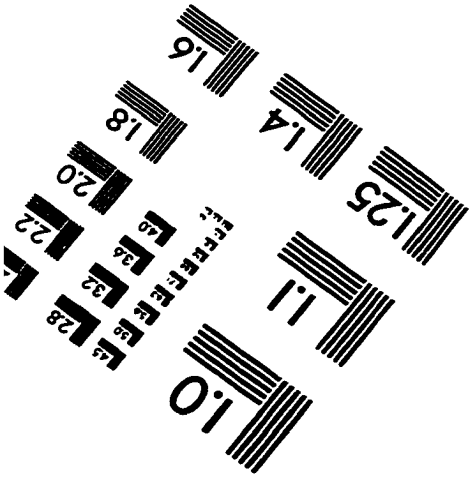
IMAGE EVALUATION TEST TARGET (QA-3)





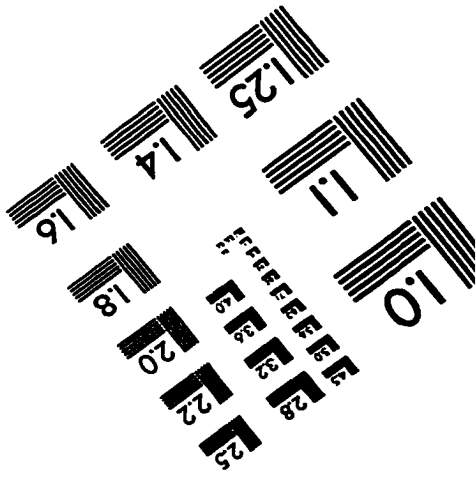






APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.