Order Number 9315493

Modelling objects, knowledge and learning in distributed object-based systems

Ndjatou, Gilbert, Ph.D.

City University of New York, 1993

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

MODELLING OBJECTS, KNOWLEDGE AND LEARNING IN DISTRIBUTED OBJECT-BASED SYSTEMS

by

Gilbert Ndjatou

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.

1993

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.

1/26/93
Date

R. Parikh
Chair of Examining Committee

1/26/93
Date

Executive Officer

Prof. Melvin Fitting

Prof. Paul Krasucki

Prof. Stanley Habib

Prof. Rohit Parikh (Chairman)

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

ABSTRACT

MODELLING OBJECTS, KNOWLEDGE AND LEARNING IN DISTRIBUTED OBJECT-BASED SYSTEMS

by

Gilbert Ndjatou

Adviser: Professor Rohit Parikh

We discuss some issues about Object-Oriented concepts, and also propose a formal model of objects and distributed object-based systems, based on that of reactive systems. Our model is intended to provide a means of specifying and reasoning about the behavior of systems consisting of active, autonomous and interacting components, as well as those of the constituting components while abstracting the detailed mechanisms involved. These systems may consist of software components, workstations, concurrent Object-Oriented programming languages constructs, individuals, ... etc. We also provide mechanisms to describe object classes and inheritance within our model. These mechanisms are behavioral and are similar to those suggested by O. Nierstrasz and M. Papathomas [NP90] or P. America and F. Van der Linden [AV90].

Unlike previous models of reactive systems, we would like ours to capture both the data and the process views of objects, on the same lines as the objectcharts of S. Bear et al [BACH90], or L. Lamport's [L89] transition axiom method. Objects are thus viewed as reactive systems with some intelligence and a decision mechanism or the organ of will of J. McCarthy and P.J Hayes [MH69]. To capture the intelligence of objects, we introduce and reason about objects' knowledge within our framework. We also introduce the notions of objects' procedural knowledge and learning in distributed object-based systems. It will then be possible to provide a knowledge-based characterization of object "similarity", inheritance and rational behavior.

An object's knowledge corresponds to facts about its task-domain situations that it should be said to know according to its behavior specifications or protocol. It is static and does not depend on system computations. It is introduced in a modal and dynamic logics framework, and leads to a logic of knowledge and actions. We also prove the soundness, completeness, and decidability of that logic. An object's procedural knowledge corresponds to the notion of "laws of ability" in [MH69], and expresses the ability of an object to use and transform its knowledge of facts about its task-domain situations. It is specified in terms of objects' knowledge and actions.

Our approach to learning is related to the work of K.M. Chandy and J. Misra [CM86], and that of R. Parikh and R. Ramanujam [PR85], and corresponds to the knowledge obtained and transfered in distributed object-based systems. It also corresponds to object instances' awareness of their own knowledge or properties of computations in distributed object-based systems, and is closely related to the intuitive notion of learning by experience. We introduce a modal logic of knowledge and learning/awareness, and also prove some properties of learning/awareness and forgetting in distributed object-based systems. This approach to knowledge and learning/awareness also gives us a semantical basis to the distinction and relationship between these two aspects of knowledge in distributed object-based systems that we refer to as static/local knowledge and dynamic knowledge. We are also hopeful that it will help to bring out a solution to some of the problems that J. McCarthy and P.J Hayes [MH69] pointed out as arising in constructing the epistemological part of an artificial intelligence.

ACKNOWLEDGEMENT

I would like to dedicate this work to the memory of my grandmother Ngatcha Lucienne, who died shortly after I left to pursue these studies. She has been an inspiration for my whole life.

I could not find the right words for my advisor and mentor, professor Rohit Parikh. His time has been countless, and his devotion is very appreciated. He introduced me to the beauty of logic and theoretical computer science, and his exigence has pushed me to the limit. I have gained great confidence by working with him. I am also very grateful for the support that he provided me through the NSF grant N° CDA-8803409.

I would like to thank the members of my doctoral committee, professors Melvin Fitting and Paul Krasucki for their time and effort in reading the manuscript of this work, and making helpful suggestions. Many discussions with Dr. Peter Barnett have been useful, and I realy appreciate that.

I am very grateful for the support that I have received from the departments of Computer Science at the College of Staten Island and Brooklyn College. Special thanks go to professors Robert Orchard, Charles Giardina, Miriam Tausner and my friend Elizabeth worthman at the College of Staten Island, and professors Stathis Zakos and Kenneth McCaloon at Brooklyn College.

Finally, special thanks to my wife Victorine, and children Edgar and Francine whom I have given little time for family life while working on this dissertation. I am grateful for their moral support and understanding.

TABLE OF CONTENTS

I- MOTIVATION AND BACKGROUND

There is a general class of systems that consist of autonomous and active entities, with relative dependencies. Such systems are characterized by a lack of global goals. Entities independently perform their tasks in a distributed environment, but occasionally interact with each other via message passing. Examples of such systems include communities of individuals, office automated systems [NT89] and groupware systems [EGR91]. In fact, most systems of entities that may qualify as intelligent also fall in this category.

Entities have behavioral patterns that are characterized at varied degrees by the following properties:

a) Knowledge and activities encapsulation: Each entity's task domain is hidden from other entities. Communication of messages is the only form of interaction. Entities' behaviors are perceived in their globality; procedural details are hidden or are not relevant, the final result being the primary concern. For example, when you take your car to a mechanic, your primary concern is that the problem goes away. You do not know (unless you are a mechanic yourself), and most of the time are not concerned with how the problem got fixed. Also, another mechanic may solve the problem in a different way, and the result will be just fine.

b) Autonomous responsibility and control: Each entity has total control over its activities, including communication with other entities. Its behavior is solely a function of its task domain situations, even though it might be affected by interactions with other entities. Note that by task-domain situations, we refer to the information that entities have about their resources, and the status of communication with other entities.

c) Property-based communication: By communication, we refer to all of the procedures by which one mind may affect another [W49]. This can be manifested in many forms. But in any case, the sender makes explicit use of a receiver's property that is accessible or available to him. When you speak to me, you make the implicit assumption that I will listen. Also, you call Peter to repair your broken TV only if you know that Peter is able to repair TVs.

We will refer to these properties as external or communication properties of entities. They are ways of doing or feeling that are accessible to and activated by other entities, and involve some activities for the receiver. They correspond to methods in object-oriented programming languages, L. Lamport's [L89] interface actions, the general sensory organs of J. McCarthy and P.J. Hayes [MH69], or I/O automata's input actions [RWZ92]. Communication is thus viewed as a causality relation among entities. Notice that although this property eliminates the semantical problem of communication [W49], it does not resolve the effectiveness problem. That means: The success with which the meaning conveyed to the receiver leads to the desired conduct on his part [W49]. For the communication, effectiveness depends on the current task domain situation and/or context of the receiver.

d) Context dependency: At any given instant, an entity's external property may be visible or accessible only to some entities in the system. Only these entities may initiate a communication (or an external action) using that property. For example, suppose I have a magical power to cure the flu. Unless I advertise my skills, only some close relatives will know it and thus may request my help when they are sick with the flu. Also, as mentioned above, the fact that they have access to my skill does not guarantee that any request for help will be satisfied. I may be busy or not willing to help.

e) Mutation: Entities are able to change or discard some of their properties, depending on their task domain situations and/or context. For example, suppose John is a mechanic who repairs brakes, transmissions, and other engine problems. At some instant, he may decide not to repair brakes any more, because there are few brake jobs or because brake repairs are less lucrative than other jobs. He may also add frame jobs because he has just hired someone who can do it.

With the increasing development of concepts in object-based programming languages and distributed systems, these types of systems have gained a great interest in computer science: As stated by C. Tomlinson and M. Scheevel [TS89], "The message-passing metaphor treats objects as autonomous active entities that synchronize and exchange information with one another only by explicitly sending messages. This view is

equivalent to stating that each object 'completely encapsulates' some local state that may be accessed only by methods that are somehow associated with the object (usually via a "class" definition). Objects may access the local state of another object only by requesting that the recipient of a message execute some method". In fact, several systems that incorporate the above entities' characteristics have been designed and implemented by combining concepts from distributed systems and object-based programming languages. An object-based programming language is characterized by features that enable its users to design and develop a program consisting of multiple interacting, autonomous components called objects, whereas a distributed operating system permits a collection of workstations or personal computers to be treated as a single entity [CC91]. However, in the above systems, the operating system must also supply a global, machine-wide object space, by providing features for object management, object interaction management, and resource management [CC91]. Some paradigms for process interactions in distributed computations are described in [An91]; and a discussion on concurrent object-oriented programming languages features and implications is found in [TS89]. They also consider parallel and distributed forms of concurrency. Also, a survey of object-oriented concepts is given by O. Nierstrasz [Ni89], and that of existing systems by R.S. Chin and S.T. Chanson [CC91] and C.A. Ellis and S.J. Gibbs [EG89]. The focus of Ellis and Gibbs is on objects' "active" aspects and issues. They also point out the need for a shift of perspective of the object-oriented paradigm from implementation to modeling. In particular, they observe that: "... the future of object-oriented paradigm is to view it as a way of thinking and doing rather than simply as a way of programming". This approach is shared by Y. Wand [Wa89] who also proposes a formal model of objects, based on M. Bunge's formalization of ontology [B77] and [B79]. His model views objects as "constructs for modeling the problem domain", and is more concerned with "passive objects": One that must receive a message before performing any action [EG89]. Several other object models have been suggested in the literature, among which are those of S. Bear et al. [BACH90] and O. Nierstrasz and M. Papathomas [NP90]. But in general, these models are implementation-

driven whereas our approach would like to address the rather general question of "how can we view the world in terms of objects?".

Our model is intended to provide a means of specifying and reasoning about the behavior of systems consisting of active, autonomous, and interacting components as well as those of the constituting components, while abstracting the detailed mechanisms involved. In this respect, objects are considered with respect to their usage and not their implementation. As observed by W. Kent [K92], "object models differ in the extent to which they differentiate between user and developer views, and the extent to which they focus on one or the other. They differ in the essential nature of objects as presented to users, and in the degree to which developer's implementation concerns are incapsulated from users". He also observes that the object user sees: An object request interface, the consequences of making requests at that interface, and the specifications describing the consequences he should expect.

Our approach is closely related to that of reactive systems, which have been largely investigated: [Ab87], [BHR84], [H87], [HPSS87], [HM85], [M80], [Pn87] and [RWZ92]. Characteristics of what might qualify as a reactive system are found in [Pn87]. They are best described by their ability to maintain an interaction with their environment. However, unlike previous models of reactive systems, we would like ours to capture both the data and the process views of objects on the same lines as the objectcharts of S. Bear et al. [BACH90], or L. Lamport's [L89] transition axiom method. "Objects" are not processes: They use processes for their control, but at the same time, they keep their identity away from processes by means of some control mechanisms. Processes are able to access or modify only parts of the object that they are allowed to access or modify. Also, modifying parts of an object does not change its identity.

Objects are reactive systems with some intelligence, or the hysteretic agents of M.R. Genesereth and N.J. Nilson [GN87]. They do not just do things; they have a decision mechanism or the organ of will of J. McCarthy and P.J. Hayes [MH69]. The performance of their activities depends upon the information that they have about their task-domain situations: You perform some activities or initiate a communication only

if you are in some need in doing so. It thus follows that in our model, a system is not identified with its set of runs, or histories, or computations as in previous models of reactive systems or knowledge [CM86], [HF89] and [PR85]. As stated in [MH69]: "A computer program capable of acting intelligently in the world must have a general representation of the world in terms of which its inputs are interpreted. Designing such a program requires commitments about what knowledge is and how it is obtained". They characterize intelligence as follows: "... an entity is intelligent if it has an adequate model of the world (including the intellectual world of mathematics, understanding of its own goals and other mental processes), if it is clever enough to answer a wide variety of questions on the basis of this model, if it can get additional information from the external world when required, and can perform such tasks in the external world as its goals demand and its physical abilities permit". They also observe that "intelligence" has two parts, namely: The "epistemological" and the "heuristic". "The epistemological part is the representation of the world in such a form that the solution of problems follows from the facts expressed in the representation. The heuristic part is the mechanism that on the basis of the information solves the problem and decides what to do".
Our model differs from their automaton representation of the world in that interconnections between entities are not fixed, and depend instead on entities' local states; communication among entities is asynchronous, and an entity can receive at most one message at a time.

In order to capture the intelligence of objects, we introduce and reason about objects' knowledge within our framework. We also introduce the notions of objects' procedural knowledge, and learning/awareness in distributed object based systems. It will then be possible to provide a knowledge-based characterization of object "similarity", inheritance and rational behavior.
An object's knowledge corresponds to facts about its task-domain situations that it should be said to know according to its behavior specification or protocol. It is static, and does not depend on system computations. Although several models of knowledge in distributed environments have been suggested, [CM86], [HF89], [Mos92], [Ne88], [Pa90] and [PR85], little attention has been given to this aspect of knowledge, despite

the fact that its importance in constructing artificial intelligence and understanding natural intelligence has been noticed by J. McCarthy and P.J. Hayes [MH69]: "... we regard the construction of intelligent machines as fact manipulators as being the best bet both for constructing artificial intelligence and understanding natural intelligence". In fact, in most formal treatments of knowledge [CM86], it is defined in terms of what we refer to as learning/awareness, which by the way depends on system computations. In our model, it is introduced in a modal and dynamic logics framework, and leads to a logic of knowledge and actions. We also prove the soundness, completeness, and decidability of that logic. An object's procedural knowledge corresponds to the notion of "laws of ability" in [MH69], and expresses the ability of an object to use and transform its knowlege of facts about its task-domain situations. It is specified in terms of objects' knowledge and actions. Our approach to learning is related to the work of K.M. Chandy and J. Misra [CM86], and that of R. Parikh and R. Ramanujam [PR85], and is called "knowledge" in [PR85]. It corresponds to the knowledge obtained and transfered in distributed object-based systems. It also corresponds to object instances' (an object given some initial state) awareness of their own knowledge or properties of computations in distributed object-based systems, and is closely related to the intuitive notion of learning by experience. We introduce a modal logic of knowledge and learning/awareness and also prove some properties of learning/awareness and forgetting in distributed object-based systems.

This approach to knowledge and learning/awareness gives us a semantical basis to the distinction and relationship between these two aspects of knowledge in distributed object-based systems that we refer to as static/local knowledge and dynamic knowledge. We are also hopeful that it will help to bring out a solution to some of the problems that J. McCarthy and P.J. Hayes [MH69] pointed out as arising in constructing the epistemological part of artificial intelligence:

1. What kind of general representation of the world will allow the incorporation of specific observations and new scientific laws as they are discovered?
2. Besides the representation of the physical world what other kind of entities have to be provided for? For example, mathematical systems, goals, states of knowledge.
3. How are observations to be used to get knowledge about the world, and how are the other kinds of knowledge to be obtained? In particular what kinds of knowledge about the system's own state of mind are to be provided for?
4. In what kind of internal notation is the system's knowledge to be expressed?

In the next section, we discuss some issues about entities' concepts, and also attempt to provide a formal characterization of those concepts. In section 3, we present our model of entities called object, and that of a distributed object-based system. We give definitions of some particular objects, such as "passive" and "active" objects within our framework, and also introduce the notion of "behaviorally indistinguishable" local states of an object as a way to observe its behavior specification or protocol in order to get some information about its procedural knowledge. Our model of system computations is also presented. In section 4, the notion of "behaviorally similar" objects (introduced in section 2) and that of "object class" and "inheritance" are formally characterized. These characterizations are behavioral, and are based on our notion of behaviorally indistinguishable local states. In section 5, we discuss some issues concerning objects' knowledge in distributed environments, and also introduce the notion of local/static knowledge. A logic of knowledge and actions is presented, and we also prove its soundness, completeness and decidability. Section 6 is concerned with the semantics of objects. A formal characterization of an object's procedural knowledge is presented, and we also provide a knowledge-based characterization of object "similarity", inheritance and rational behavior. In section 7, we discuss some issues about learning and awareness in distributed object-based systems. We then introduce a modal logic of knowledge and learning/awareness, and also prove some properties of learning/awareness and forgetting.

2- ENTITIES CONCEPTS

In this section, we will attempt to formally characterize entities, by introducing the notions of local states, primitive actions, context, and external properties. We will also introduce the notions of systems computations, indistinguishable local states and entities' similarities. We allow entities to be non-deterministic: By this, we mean that entities might have more than one way to perform their activities or might have a choice of activities at any state. Our system of reference will consist of a set $I = \{ 1, 2, ..., n\}$ of entities, with $n \geq 2$.

2.1 LOCAL STATES - PRIMITIVE ACTIONS

Entities' objective is to perform some tasks, by executing an appropriate sequence of activities (procedural or communication initiation). However, they may be interrupted from time to time by communication from other entities. At any given instant, knowledge about an entity's task-domain situation is described by its local state. (Note that by task-domain situations, we refer to the information that entities have about their resources, and the status of communication with other entities). It consists of data structures or any form of representation suitable for such knowledge. As suggested by W. Kent [K92], it might be described in terms of variables included in the object descriptions in a programming environment. The knowledge encapsulation property of entities implies that: if Q^i is the set of local states of entity i , then $Q^i \cap Q^j = \emptyset$ for $i \neq j$. This corresponds to the nonshared variable model of Object-Oriented programming languages.

Each entity is characterized by the procedural activities it can or cannot perform. We will refer to these activities as primitive actions, and thus assign to each entity i a set A^i of its primitive actions. Primitive actions correspond to I/O automata's internal actions. The reason why we use the term "primitive" will become clear shortly. Note that procedural activities of entities are observed through their effects on the task-domain situations.

Since we are not concerned with or do not have access to procedural details of such activities, only the local states before and after their execution are relevant. So, to each primitive action a of entity i corresponds a relation $R^i_a \subseteq Q^i \times Q^i$.
R^i_a is the set of transition states corresponding to action a for entity i.
Intuitively , $(s,t) \in R^i_a$ if and only if:

i) it is possible for entity i to perform primitive action a in state s and,

ii) executing primitive action a in state s may cause entity i to move to state t.

For $s \in Q^i$ and $a \in A^i$, we will assume that the set $\{t \mid (s,t) \in R^i_a\}$ is finite. This property is referred to as image-finiteness in [HM85]. We also assume that primitive actions are atomic actions.

2.2 EXTERNAL PROPERTIES AND ENTITIES' CONTEXT

An entity's external properties corresponds to methods in object-oriented programming languages, L. Lamport's [L89] interface actions, I/O automata's input actions [RWZ92], or general sensory organs [MH69]. As explained in the above section, entities communicate with each other by making explicit use of some receiver's external property that is available or accessible to them. For an entity i's external property e, we will say that an entity $j \neq i$ is in the scope of e with respect to i if and only if there is a local state of entity j in which e is available or accessible to j for communication to i. Notice that our notion of scope is similar to that of programming languages.
For example, suppose that $I = \{ 1, 2, 3 \}$ and $E^1 = \{ e, e' \}$, $E^2 = \{ e' \}$, $E^3 = \emptyset$ are the sets of external properties of entities 1 , 2 , and 3 respectively. Suppose also that: Entities 1 and 3 communicate with entity 2 using e'; entity 2 communicates with entity 1 using e and entity 3 communicates with entity 1 using e'. Then: entities 1 and 3 are in the scope of e' with respect to entity 2 , entity 2 is in the scope of e with respect to entity 1 and entity 3 is in the scope of e' with respect to entity 1.

We define the context of an entity j to be the set of all pairs (e,i) such that j is in the scope of e with respect to i. In other words, the context of an entity is the set of all possible communications available or accessible to it.

It describes at the same time the types of communication and the potential receivers. It also specifies the information that an entity has about its environment. It corresponds to the notion of "required interface" in object-oriented programming languages, whereas an entity's set of external properties corresponds to that of "client or provided interface" [BACH90]. A similar notion called "capability scheme" in [CC91] is used as a security mechanism in DOBPS. So, the context of an entity i is the set
$C^i = \{ (e,j) \mid i \text{ is in the scope of } e \text{ with respect to } j \}$ and $(e,j) \in C^i$ if and only if there is a state s in Q^i where it is possible for i to initiate a communication to j using j's external property e. In the above example, we have:
$C^1 = \{ (e',2) \}$, $C^2 = \{ (e,1) \}$ and $C^3 = \{ (e',2), (e',1) \}$.

2.3 EXTERNAL PROPERTIES AND COMMUNICATIONS

Unlike primitive action executions, communication initiations do not guarantee that communication will effectively take place. Whether a communication effectively takes place depends on the receiver's current local state and activity, which the sender may not know because of the knowledge and activities encapsulation property of entities. As far as the sender is concerned, a communication initiation may or may not result in the receiver being affected in the desired way, the observable effect being some state transition of the receiver. Although the sender need not know what really happens at the receiver's task-domain, it may in turn be affected as a result of what did happen there, whether the communication did effectively take place or not. To illustrate our point, consider the example of a student preparing his final exam who needs help from his instructor to solve a particular problem. He then walks to his instructor's office and knocks at the door. What happens next will depend on the instructor's mood and preocupations at this instant. With no pretension to be exhaustive, any of the following situations may result:

1) The instructor gets up, opens the door, listens to the student, solves the problem, and then explains it to the student.
2) He gets up, opens the door, listens to the student, and then for some reason, refuses to help.
3) He has stepped out of the office for a cup of coffee.
4) He is in a bad mood and does not want to be disturbed.

The communication did effectively take place in the first two cases (i.e the instructor did receive the message), involving a sequence of activities and then a communication initiation by the instructor to the student. Notice that it need not be the case that any communication that effectively takes place must result in some communication initiation by the receiver to the sender. When this is the case, we will consider it as a sequence of two different events. With this consideration, we stipulate that the sender is not affected by communication initiations (i.e. its local state does not change after an external action or a communication initation). Communication is thus viewed as a causality relation among entities: As stated in [MH69], "... the whole idea of a system of interacting automata is just a formalization of the commonsense notion of causality". Also, in the ideal object-oriented model, each object maintains its own local state that is accessible only by its own local methods. All other (external) access are achieved via message passing [TS89].

Notice that in the first case, the communication has been effective in the sense of [W49] whereas it is not in the second one. A communication that effectively takes place may involve more than one activity for the receiver as in the above example. But since we assume the encapsulation of activities, the relevant effect will be a state transition of the receiver. In our particular example, the instructor may have used other means to open the door (without getting up) or to get a solution to the problem, and the effect might have been the same state transitions as in the above cases.

From the above discussions, it becomes clear that to each entity i's external property e corresponds a relation $S^i_e \subseteq \bigcup_{j \neq i}(Q^j \times Q^i \times Q^j)$.

In other words, the state of an entity after the receipt of a message (or the execution of an external action) depends on its current state, and the state of the entity that initiated the communication (or action). This approach to entities interactions is also used in [MH69]. In fact, as they suggested, for each entity in the system, a message will correspond to a class of its local states that convey the same information to all other entities. Local states u, v of entity j are said to convey the same information to entity i with respect to external property e at s if for each local state t of i, $(u, s,t) \in S^i_e$ iff $(v, s, t) \in S^i_e$.

Intuitively, for $u \in Q^j$, and $s, t \in Q^i$, $(u,s,t) \in S^i_e$ iff:

i) In state u, entity j may initiate a communication (or an external action) to i with respect to external property e, and

ii) If it does it, i may receive the message (or execute the external action) in state s, with resulting effect the transition to t.

For $e \in E^i$ and $s \in Q^i$, we will also require that the set $\{t \mid \exists v \in \bigcup_{j \neq i} Q^j, (v,s,t) \in S^i_e\}$ be finite.

For $e \in E^i$ and $s \in Q^i$, the transitions corresponding to external property e at s do not depend on the identity of the entity initiating the communication (or external action), but only on the types of information conveyed to i at s.

In the last two cases of the above example, the communication did not effectively take place (i.e. the instructor did not receive the message). However, if we assume that the instructor has a mail box in front of his office where students can leave their messages, then it will still be possible for the instructor to get the student's message at a later time. It thus follows that interactions among entities are of the type asynchronous message passing.

2.4 EXAMPLE: THE SODA MACHINE SYSTEM

The soda machine system consists of 4 entities:

1. the soda machine
2. the cans container (with two types of soda)
3. a lamp with two light bulbs (with each light bulb indicating that there is no soda of some type, when the light is on)
4. the user

LOCAL STATES

Soda machine: consist of the tuples (T-AMOUNT, D-AMOUNT, SELECTION) where: T-AMOUNT = 60n , $n \geq 0$ is the total amount received so far; D-AMOUNT = 5n , $0 \leq n \leq 12$ is the amount currently being processed, and SELECTION = 0 for no selection; 1 for soda #1; and 2 for soda #2 corresponds to the selection made by the user.

Cans container: consist of the tuples (#CAN1, #CAN2, OPEN1, OPEN2) where:
#CAN1 = number of soda #1 cans; #CAN2 = number of soda #2 cans
OPEN1 = 0 (for gate of soda #1 closed) and 1 (for gate of soda #1 opened)
OPEN2 = 0 (for gate of soda #2 closed) and 1 (for gate of soda #2 opened)

Lamp: consist of the tuples (light #1 , light #2) where:
light #1 = ON (no more soda #1) and OFF otherwise
light #2 = ON (no more soda #2) and OFF otherwise

User: consists of the tuples (T-DEPOSIT, VALUE-COIN, CHOICE, SODA) where:
T-DEPOSIT = 5n , $0 \leq n \leq 12$ is the total amount deposited so far;
VALUE-COIN = 0 , 5 , 10 or 25 is the value of the selected coin;
CHOICE = 0 (no choice) 1 (soda #1) 2 (soda #2) 3 (give up) and
SODA = 0 (want soda) 1 (have soda #1) 2 (have soda #2).

PRIMITIVE ACTIONS

Soda machine: RESUME (resumes the sale of soda operation)

RESUME :

(T-AMOUNT, 60, S) , S = 1 | 2 ----------> (T-AMOUNT + 60, 0, 0)

Cans container: UPDATE (closes the opened gate and updates the number of soda cans)

UPDATE:

(#CAN1, #CAN2, 1, 0) , #CAN1 $\neq$ 0 ----------> (#CAN1 - 1, #CAN2, 0, 0)

(#CAN1, #CAN2, 0, 1) , #CAN2 $\neq$ 0 ----------> (#CAN1, #CAN2 - 1, 0, 0)

User: SEARCH-COIN (gets a coin from its pocket) and

MAKE-CHOICE (makes up its mind about the type of soda it wants)

SEARCH-COIN:

(T-DEPOSIT, V-COIN, 0, 0) ----------> (T-DEPOSIT + V-COIN, V', 0 , 0)

T-DEPOSIT + V-COIN < 60 V' = 5, 10 OR 25 s.t.

T-DEPOSIT + V-COIN + V' $\leq$ 60

MAKE-CHOICE:

(T-DEPOSIT, V-COIN, 0, 0) ----------> (60 , 0 , C , 0) , C = 1 | 2

T-DEPOSIT + V-COIN = 60

EXTERNAL PROPERTIES

Soda machine:

EXTERNAL PROPERTIES	CLIENTS
GET-COIN	USER
GET-SELECTION	USER

GET-COIN:

(T-DEPOSIT, V-COIN, 0, 0) .....> (T-AMOUNT, D-AMOUNT, 0)

T-DEPOSIT + V-COIN ≤ 60 T-DEPOSIT = D-AMOUNT

(T-AMOUNT, D-AMOUNT + V-COIN , 0)

GET-SELECTION:

(60, 0, C, 0), C = 1 | 2 .....> (T-AMOUNT, 60, 0)

(T-AMOUNT, 60, C)

Cans container:

EXTERNAL PROPERTIES	CLIENTS
RELEASE-CAN	SODA MACHINE

RELEASE-CAN:

(T-AMOUNT, 60, 1) .....> (#CAN1, #CAN2, 0, 0)

(#CAN1, #CAN2, 1, 0)

(T-AMOUNT, 60, 2) .....> (#CAN1, #CAN2, 0, 0)

(#CAN1, #CAN2, 0, 1)

Lamp:

EXTERNAL PROPERTIES	CLIENTS
DISPLAY-LIGHT	CANS CONTAINER

DISPLAY-LIGHT:

(0, #CAN2, 0, 0) .....> (OFF, S) ———> (ON, S)

(#CAN1, 0, 0, 0) .....> (S, OFF) ———> (S, ON)

User:

EXTERNAL PROPERTIES	CLIENTS
SEE-LIGHT	LAMP
GET-CAN	CANS CONTAINER

SEE-LIGHT:

(ON, ON) .....> (0, 0, 0, 0) ————> (0, 0, 0, 3)

(ON, OFF) .....> (T-DEPOSIT, V-COIN, 0, 0)

T-DEPOSIT + V-COIN = 60

↓

(60, 0, 2, 0)

(OFF, ON) .....> (T-DEPOSIT, V-COIN, 0, 0)

T-DEPOSIT + V-COIN = 60

↓

(60, 0, 1, 0)

GET-CAN:

(#CAN1, #CAN2, 1, 0) .....> (60, 0, 1, 0)

#CAN1 ≠ 0

↓

(60, 0, 1, 1)

(#CAN1, #CAN2, 0, 1) .....> (60, 0, 2, 0)

#CAN2 ≠ 0

↓

(60, 0, 2, 2)

In the above example, the user goes to the soda machine when (s)he wants the soda. However, if both lights are on (i.e. there is no soda), (s)he walks away. Otherwise, (s)he gets coins (nickels, dimes or quarters) from his/her pocket and deposits them into

the soda machine until the amount deposited equals 60 cents (the price of a soda can). At this point, (s)he makes up his/her mind about the type of soda (s)he wants, and then makes his/her selection. For simplicity, the soda machine does not accept more than 60 cents per sale operation (it does not return change). Also, if one of the lights is on (i.e. there is only one type of soda), this type of soda becomes the choice of the user. We also assume for simplicity, that the user always makes up his/her mind about the type of soda (s)he wants after (s)he has already deposited 60 cents.

After the soda machine has received 60 cents and the selection of soda, it causes the cans container to release a can of the soda selected, and then resumes the sale operation. After the cans container has released a can of soda, it updates the number of cans of that type of soda and, if it becomes zero, it causes the lamp to display the appropriate light.

2.5 SYSTEMS OF ENTITIES AND COMPUTATIONS

For each entity in the system, the transitions corresponding to its primitive actions and the communication initiations with their corresponding transitions describe what the entity can do locally (i.e. its protocol) and the corresponding local and external effects. (Note that by protocol, we refer to the same notion as in [HF89]). This corresponds to L. Lamport's [L89] safety properties of the system. However, as he suggested, a system specification must also describe what the system must do. I.e. its liveness properties. Although some of these properties are implied by the safety properties, others are not and must be specified explicitly. A detailed discussion of some of the issues about systems computations is also found in [HF89]. In fact, they observe that "... in practice, the set of runs making up the system will be chosen by the system designer or the person analyzing the system, who presumably has a model of what the possible executions of the protocol are".

In our particular context (note that we are specifying a class of systems) these properties may be characterized as "general" and "specific". A general property is one that must be true for all systems whereas a property will be said to be specific if it is true only for some systems.

The first of our general properties is more an assumption and asserts that each basic ativity must be completed in unit time (with time ranging over the natural numbers); the second asserts that a message is received only if it has been sent earlier; and the third states that messages do not have to be received in the same order in which they have been sent. There is also the possibility that a message sent is never received. Note that these properties have been previously used in other models [CM86], [HF89] and [MH69].

"Specific" properties are more concerned with interactions between entities: For instance, in a system with a bounded buffer, the number of messages sent but not yet received must always be less than or equal to the size of the buffer. Also, in most object-oriented programming languages, an object must send a message only once, and then either proceed with the execution of a primitive action, or is blocked until it receives a message (if it is accesssible at that state) and otherwise, is blocked indefinitely. On the other hand, there are systems for which this is not true. In particular, systems involving humans. For instance, in the "student-instructor" example above, the student could continue to knock at the instructor's office door until he receives an answer or until he is interrupted by another event. In fact, there are many situations in real life where people tend to repeat their actions, especially when they do not get an answer. For example, a child talking to his mother, telephone calls, ...etc.

Note also that it is possible to specify as a safety condition the maximum number of times a message can be sent without an answer. This may be done by using a counter or a local clock that is incremented each time a new call or request is not followed by an answer.

It thus follows from the above discussions that there is no unique characterization of systems computations. However, for the purpose of our theory, it will be enough to know that they satisfy the "general" liveness properties.

2.6 INDISTINGUISHABLE LOCAL STATES

Entities perform their tasks, using two different kinds of knowledge: First, the declarative knowledge or information embodied in their local states and second, their procedural knowledge (how entities use and transform their declarative knowledge). Because of the encapsulation property of entities, the only information that we have about their procedural knowledge is through their local states' transitions. An entity's local state conveys information about the entity's procedural knowledge in two different ways: first, the possibility of the entity to initiate some types of communication in that state, and second, the different ways in which it is transformed to other local states.

Intuitively, two local states of an entity will be said to be behaviorally indistinguishable if they convey the same information about the entity's procedural knowledge. In other words, the entity can only initiate the same types of communication in both states, and the different transformations of one by the entity do not tell us more about the entity's procedural knowledge than those of the other.

For an observer, their differences are irrelevant to the entity's protocol. For example, suppose there is a physician who always asks his patients to take two aspirin tablets with some liquid that he gives them every time they go to him with a foot pain, a tooth-ache , or a stomach-ache; and every time they follow his instructions, they feel better. Assuming that each medication or medication combination is used for only one illness, the least an observer may guess about his medical practice is that all these pains are symptoms of the same illness. In this case, an illness corresponds to a class of symptoms indistinguishable by medications. Observe that this concept is very similar to R. Milner's [HM85] observational equivalence. It is also closely related to the "see function" of hysteretic agents in [GN87]. Our formalization is given in the next section.

2.7 ENTITIES' SIMILARITIES

In the "student-instructor" example above, the communication consists of a request for help, and involves a certain number of activities for the instructor. Two different instructors may have acted differently even in "identical" circumstances. This is due in part to the fact that things are rarely the same for different individuals.

Something that I may find good may not be good for you. You may also find something relevant to some situation that I do not: Our perception of the world is subject to many factors that are relative to each individual. See [Pa92] and [MH69] for further discussions on these matters. This implies that entities' task-domain situations (information that entities have about their resources and the status of communication with other entities) are never the same, even though they might have similar behaviors with respect to their task-domain situations.

Entities with similar behaviors with respect to their task-domain situations will be said to be behaviorally similar. This corresponds to the situation in which the difference of their task-domain situations is irrelevant to their behaviors. It also corresponds to the notion of objects' type discussed in [AV90] and [NP90]. A more suitable example is that of an "IBM XT" and an "IBM AT", if we consider the set of operations available on the "IBM XT". Except for the speed, the extra power of the "IBM AT" is irrelevant to the performance of those operations. This notion and a related notion of inheritance are formally characterized in section 4.

Another point we would like to mention is that of the influence of the environment on entities' behaviors. In the "student- instructor" example, even the same instructor's behavior may differ in a different circumstance. For example, he would have not stepped out for a cup of coffee if he had a coffee maker in his office, or he would have not been in a bad mood if he was in a different institution with generous salaries and nice offices. This suggests that entities' behaviors are subject to their context. So, even two behaviorally similar entities may actually behave differently if they have different contexts. We will not discuss this notion further in the present work. We postpone its treatment to a later time.

3. FORMALIZATION

In this section, we introduce our formal model of entities called object, and that of a distributed object-based system (DOBS). We will also provide a formal characterization of active and passive objects, object instance, and also introduce other subcategories of objects, namely: "autonomous", "private", "stand-alone", and "null" objects. Our formalization of the notion of "behaviorally indistinguishable" local states and our model of systems computations are also presented. We start with a brief description of the subject of our model. The reader is referred to section 2 for ample discussions.

Entities are defined with respect to their environment, and are characterized by their local states, primitive actions, external properties, and their context. To each entity i correspond a nonempty set of possible local states Q^i , a set of primitive actions (containing the null action) from an alphabet of primitive actions A , and a set of external properties from an alphabet of external properties E. It is assumed that both alphabets are given and are disjoint.

Knowledge (or information) about an entity's task-domain is described by its local state. It follows from the encapsulation property discussed in the previous sections that any two distinct entities have disjoint sets of local states.

An entity's primitive action specifies a basic activity that it can independently perform, with the resulting effect a transition of its local state. To each entity's primitive action will then correspond a transition relation of its local states.

An entity's external properties correspond to L. Lamport's [L89] interface actions, methods in object-oriented programming languages, I/O automata's input actions [RWZ92], or the general sensory organs of J. McCarthy and P.J. Hayes [MH69].

The context of an entity is a set of pairs consisting of another entity in the system and one of its external properties. It specifies the information that an entity has about its environment. It also describes the types of communication and the potential receivers. So, an entity with an empty context has no information about its environment and thus, could not initiate any communication or external action.

If we denote by C^i the context of an entity i , then $(e,j) \in C^i$ if and only if entity i has information about entity j having the external property e , and thus may initiate a communication (or an external action) at some of its local states to j with respect to e. Interactions between an entity and its environment are viewed as causality relations.

An entity initiates a communication (or an external action) to another entity specified in its context by making explicit use of an external property associated with that entity. However, a communication may not effectively take place (i.e the target entity may not receive the sent message). But when it effectively takes place, the resulting effect is a state transition of the receiving entity. So, to each entity's external property corresponds a transition relation consisting of triplets (w,s,t) where s and t are local states of the entity and w is the local state of another entity which is a potential communication (or external action) initiator. It is assumed that at any local state, the transitions corresponding to an external property do not depend on the entity initiating the communication, but only on the types of information conveyed at that state. In fact, as in [MH69], we identify the messages with the local states from which they are sent, and two different messages may convey the same information (with respect to an external property) to a receiving entity at some local state.

Local states w and u will be said to convey the same information to entity i at s with respect to external property e if for any local state t of i , (w,s,t) is an e-transition if and only if (u,s,t) is. As suggested in [MH69], for each entity, a message corresponds to a class of its local states that convey the same information to all other entities in the system. These transition relations, as well as those corresponding to primitive actions must also verify the image-finiteness property (defined in section 2.1) at any state $s \in Q^i$. It is also assumed that interactions between entities are asynchronous.

NOTATIONS: If R is a subset of $\cup_i(A \times Q^i \times Q^i)$ and

S a subset of $\cup_i(\cup_{j \neq i}(E \times Q^i \times Q^i \times Q^j)$, then we denote by:

R^i_a the set $\{ (s,t) \mid (s,t) \in Q^i \times Q^i, (a,s,t) \in R \}$

S^i_e the set $\{ (u,s,t) \mid (s,t) \in Q^i \times Q^i, (e,u,s,t) \in S \}$ and

$S^i_{e^-}$ the set $\{ (s,u,v) \mid s \in Q^i , (e,s,u,v) \in S \}$.

Intuitively, R^i_a is the transition relation corresponding to the primitive action a of entity i, and S^i_e the transition relation corresponding to its external property e. $S^i_{\bar{e}}$ is the transition relation corresponding to the external property e of entity i's environment.

3.1 OBJECTS AND DISTRIBUTED OBJECT-BASED SYSTEMS

DEFINITION 3.1.1: Distributed Object-Based System

Given an alphabet of primitive actions A (containing the null action ϵ) and an alphabet of external properties E s.t. $A \cap E = \emptyset$ and $n \geq 2$, a Distributed Object-Based System (DOBS) is a tuple

$$O = (A , E , \{ Q^1 , \ldots , Q^n \} , R , S)$$

where:

Q^i is a nonempty set and $Q^i \cap Q^j = \emptyset$ for $i \neq j$

$R \subseteq \cup_i (A \times Q^i \times Q^i)$ s.t.

i) $(\epsilon,s,t) \in R$ iff $s = t$

ii) $\forall i \geq 1 , s \in Q^i , a \in A$, the set $\{ t \mid (a,s,t) \in R \}$ is finite

$S \subseteq \cup_i (\cup_{j \neq i} (E \times Q^j \times Q^i \times Q^j)$ s.t.

$\forall i \geq 1 , s \in Q^i , e \in E$, the set $\{ t \mid \exists v , (e,v,s,t) \in S \}$ is finite

An object O^i of the distributed object-based system O is the tuple

$$(Q^i , \{ R^i_a : a \in A \}, \{ S^i_e : e \in E \})$$

Q^i is its set of local states , $A^i = \{ a \mid a \in A , R^i_a \neq \emptyset \}$ its set of primitive actions, and $E^i = \{ e \mid e \in E , S^i_e \neq \emptyset \}$ its set of external properties.

We will assume that for each $i \geq 1$, A^i and E^i are finite.

The context of object O^i is the set

$$\{ (e, j) \mid e \in E , (s, u, v) \in S^j_e \text{ for some } s \in Q^i \text{ and } u, v \in Q^j \}$$

Objects specify all possible behaviors or protocols of entities with respect to their task-domain situations and environments. The actual behavior of an entity depends on the local state in which it is first considered. We will refer to this state as an initial state, and an object O^i given some initial state will be referred to as an object instance.

DEFINITION 3.1.2

An object instance of a distributed object-based system O is a pair (O^i, q^i_0) where O^i is an object of O, and q^i_0 a local state of O^i.

Note that the above descriptions give us a model of a DOBS. However, there are situations where one might be interested only in the behavior of a single entity with respect to its environment. In this case, there is no need to differentiate the other entities in its environment, and what they can do internally is of no interest. This approach is similar to that of hysteretic agents in M.R. Genesereth and N.J. Nilsson [GN87], or L. Lamport's [L89] transition axiom method. We then have the following definition.

DEFINITION 3.1.3: Object System

Given an alphabet A of primitive actions containing the null action ϵ and an alphabet E of external properties such that $A \cap E = \emptyset$, an Object System is a tuple:

$$O = (Q , Q\_ , \{ R_a , a \in A \} , \{ S_e , e \in E \} , \{ S_e\text{-} , e \in E \})$$

Where:

Q is the set of local states of the entity , $Q\_$ is the set of states of the environment and: $Q \neq \emptyset$, $Q\_ \neq \emptyset$ and $Q \cap Q\_ = \emptyset$.

$R_a \subseteq Q \times Q$, $a \in A$

with $R_a = \emptyset$ if a is not a primitive action of the entity.

$S_e \subseteq Q\_ \times Q \times Q$, $e \in E$ is the transition relation corresponding to the effect of the environment's action (or communication) initiation to the entity with respect to e.

$S_e\text{-} \subseteq Q \times Q\_ \times Q\_$, $e \in E$ is the transition relation corresponding to the effect of the entity's action (or communication) initiation to the environment with respect to e.

$e \in E$ is an external property of the entity if $S_e \neq \emptyset$

It is an external property of the environment if $S_e\text{-} \neq \emptyset$

The set $\{ e \mid e \in E , S_e \neq \emptyset \}$ corresponds to the "provided" (or "client") interface in object-oriented programming languages, and the set $\{ e \mid e \in E , S_e\text{-} \neq \emptyset \}$ corresponds to the "required" interface [BACH90].

Note that if $O = (A , E , \{ Q^1 , \ldots , Q^n \} , R , S)$ is a DOBS , then for each object O^i of O corresponds the object system

$$(Q^i , \cup_{j \neq i} Q^j , \{ R^i_a : a \in A \} , \{ S^i_e : e \in E \} , \{ S^i_e\text{-} : e \in E \})$$

3.2 CHARACTERIZATION OF SOME PARTICULAR OBJECTS

Object-oriented programming languages are classified by the types of objects they implement [EGR91] and [Ni89]. There are essentially two major categories of objects: "passive" and "active" objects. In [EGR91], passive objects are referred to as the ones "that must receive a message before performing any action", whereas active objects are those "in which a high degree of autonomous responsibility and control is vested". An active object is considered within its usage as "an independent agent, and frequently a source of knowledge and activities". Notice that these characteristics of active objects correspond closely to those of entities described in the previous sections. We will introduce other subcategories of objects namely: "autonomous", "private", "stand-alone", and "null" objects.

An autonomous object is one that has no information about its environment, and thus does not initiate any communication.

A private object is one that is not accessible to other objects for communication. Its existence is not known to other objects.

An object is stand-alone if it is autonomous and private. Stand-alone objects do not interact with their environment. One such object is a program module with no subroutines. We will refer to objects which are passive and stand-alone as null objects. Null objects correspond to inactive and inaccessible entities. It is apparant that these entities are "behaviorally similar", regardless of their respective task-domains.

DEFINITION 3.2.1

Given a DOBS $O = (A , E , \{ Q^1 , \ldots , Q^n \} , R , S)$, an object O^i of O is passive if $\forall a \neq \epsilon$, $R^i_a = \emptyset$, otherwise it is active.

It is autonomous if $\forall e \in E$, $S^i_e\text{-} = \emptyset$, and private if $\forall e \in E$, $S^i_e = \emptyset$

It is stand-alone if it is autonomous and private.

It is null if it is passive and stand-alone.

3.3 BEHAVIORALLY INDISTINGUISHABLE LOCAL STATES

In section 2.6, we introduced the notion of behaviorally indistinguishable local states of an entity as a way to observe its behavior specification or protocol in order to get some information about its procedural knowledge. It is assumed as in [GN87] that an entity's protocol (i.e. what it can do at any local state) depends upon its perception of the world. Two local states of an entity are said to be behaviorally indistinguishable if for an observer, their differences are irrelevant to the entity's protocol. In other words, the entity can only initiate the same types of communications in both states, and the different transformations of one by the entity do not tell us more about the entity's procedural knowledge than those of the other. We also pointed out that this notion is very similar to R. Milner's [HM85] observational equivalence, and is related to the "see function" of hysteretic agents in [GN87].

Formally , let $O = (Q , Q\_ , \{R_a , a \in A\}, \{S_e , e \in E\}, \{S_e\text{-} , e \in E\})$ be an object system. For local states $s,t \in Q$, we define the decreasing sequence of equivalence relations $\sim_n$ over Q $(n \geq 0)$ as follows:

$s \sim_0 t$ iff $\forall e \in E$ and $w,u \in Q\_$, $(s,w,u) \in S_e\text{-}$ iff $(t,w,u) \in S_e\text{-}$

$\forall n \geq 1$, $s \sim_n t$ iff

1) $\forall a \in A$, if $\exists u \in Q$ s.t. $(s,u) \in R_a$, then $\exists u' \in Q$ s.t. $u' \sim_{n-1} u$ and $(t,u') \in R_a$, and vice versa
2) $\forall e \in E$, $w \in Q\_$, if $\exists u \in Q$ s.t. $(w,s,u) \in S_e$, then $\exists u' \in Q$ s.t. $u' \sim_{n-1} u$ and $(w,t,u') \in S_e$, and vice versa

$s \sim_0 t$ iff the object may initiate a communication (or an external action) at s if and only if it may do so at t, and the same information is conveyed at both states.
It also follows from condition 2) that the effects of the actions initiated by the environment are "similar" at both states. Since $\epsilon \in A$, it follows from condition 1) that if $s \sim_k t$ then $\forall j \leq k$, $s \sim_j t$.

DEFINITION 3.3.1
s, t are behaviorally indistinguishable (denoted by $s \sim_Q t$) iff $\forall n \geq 0$, $s \sim_n t$.

Let Δ denote the subset of QxQ s.t. $(s,t) \in \Delta$ iff $s \sim_0 t$
and F the function $F : \mathcal{P}(\Delta) \dashrightarrow \mathcal{P}(\Delta)$ such that $\forall P \subseteq \Delta$, $(s,t) \in F(P)$ iff :

1) $\forall a \in A$, if $\exists u \in Q$ s.t. $(s,u) \in R_a$, then $\exists u' \in Q$ s.t. $(u',u) \in P$ and $(t,u') \in R_a$, and vice versa
2) $\forall e \in E$, $w \in Q$, if $\exists u \in Q$ s.t. $(w,s,u) \in S_e$, then $\exists u' \in Q$ s.t. $(u',u) \in P$ and $(w,t,u') \in S_e$, and vice versa

Notice that $\sim_Q = \bigcap_{n \geq 0} F^n(\Delta)$ where $F^0(\Delta) = \Delta$ and for $n \geq 1$, $F^n(\Delta) = F(F^{n-1}(\Delta))$. It thus follows from the classical fixed-point theory that if F is anticontinuous, then $\sim_Q$ is the maximum fixed-point of F.

THEOREM 3.3.1
The relation $\sim_Q$ is the maximum fixed-point of the function F.

PROOF: The proof is similar to that suggested by [HM85].

PROPOSITION 3.3.1
The behaviorally indistinguishable relation $\sim_Q$ is an equivalence relation.

PROOF: $\sim_Q$ is the limit of a decreasing sequence of equivalence relations

The equivalence class of $s \in Q$ is denoted by $\bar{s}$ and the set of all equivalence classes by $\bar{Q}$. The following property is easily proved.

PROPOSITION 3.3.2

The local states of null objects are behaviorally indistinguishable.

PROPOSITION 3.3.3

The equivalence classes are stable with respect to the relations R_a, $a \in A$ and S_e, $e \in E$ and communication (or external actions) initiations in the sense that:

a) $\forall a \in A$, $e \in E$, if $(s,t) \in R_a$ (respectively $(w,s,t) \in S_e$) then $\forall s' \in \bar{s}$, $\exists t' \in \bar{t}$ such that $(s',t') \in R_a$ (respectively $(w,s',t') \in S_e$).

b) $\forall e \in E$, the object in state s may initiate a communication to (or an action on) its environment with respect to e if and only if in any state $s' \in \bar{s}$, it may do the same. Morever, $(s,w,u) \in S_{e^-}$ iff $\forall s' \in \bar{s}$, $(s',w,u) \in S_{e^-}$.

PROOF: a) follows from properties 1) and 2) of the definition of $\sim_n$, $n \geq 1$ and b) follows from the definition of $\sim_0$ and the fact that $\sim_Q$ is a subset of $\sim_0$.

From the above proposition, it follows that the relations R_a and S_e can be extended to relations $\bar{R}_a$ and $\bar{S}_e$ on $\bar{Q}$ as follows:

$$(\bar{s},\bar{t}) \in \bar{R}_a \quad \text{iff} \quad \exists t' \in \bar{t},\ (s,t') \in R_a \quad \text{and}$$
$$(w,\bar{s},\bar{t}) \in \bar{S}_e \quad \text{iff} \quad \exists t' \in \bar{t},\ (w, s,t') \in S_e .$$

$(\bar{s},\bar{t}) \in \bar{R}_a$ states that at $s' \in \bar{s}$, there is an execution of the primitive action a that terminates at some local state in $\bar{t}$, and all executions of a at s' that terminate in some local state in $\bar{t}$ are indistinguishable.

$(w,\bar{s},\bar{t}) \in \bar{S}_e$ states that at $s' \in \bar{s}$, it is possible for the object to receive a message with respect to external property e that will cause it to move to some state $t' \in \bar{t}$ and that all messages w' such that $(w',\bar{s},\bar{t}) \in \bar{S}_e$ are indistinguishable at s' with respect to e.

REMARK: It follows from proposition 3.3.3b) that for each $s \in Q$, the message sent at s corresponds to the equivalence class of s, $\bar{s}$.

Note also that in the usual sense of "know", we often say that we know someone only if we can predict its behavior. In other words, if we know its state partition under the behaviorally indistinguishable relation (although it is not always the case). It thus follows that the transitions corresponding to the relations $\bar{R}_a$ and $\bar{S}_\epsilon$, and the possibilities to initiate communications at states in some equivalence classes are the least one could be said to know about the procedural knowledge of the object.

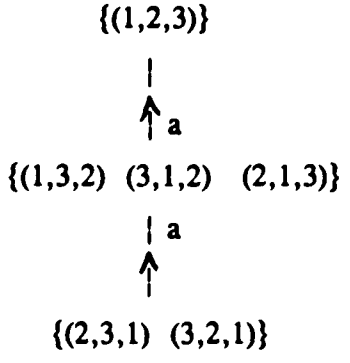
EXAMPLE 3.3.1

Let O^1 denote the stand-alone object with six local states such that:

$Q^1 = \{ (1,2,3), (2,1,3), (3,1,2), (3,2,1), (1,3,2), (2,3,1) \}$, $A^1 = \{ \epsilon, a \}$ with

$R^1_a = \{ [(1,3,2), (1,2,3)], [(3,1,2), (1,2,3)], [(2,1,3), (1,2,3)], [(3,2,1), (2,1,3)], [(2,3,1), (2,1,3)] \}$.

The equivalence classes are: $\{(1,2,3)\}$, $\{(1,3,2), (3,1,2), (2,1,3)\}$ and $\{(2,3,1), (3,2,1)\}$, and the relation $\bar{R}^1_a$ is defined by the following diagram:



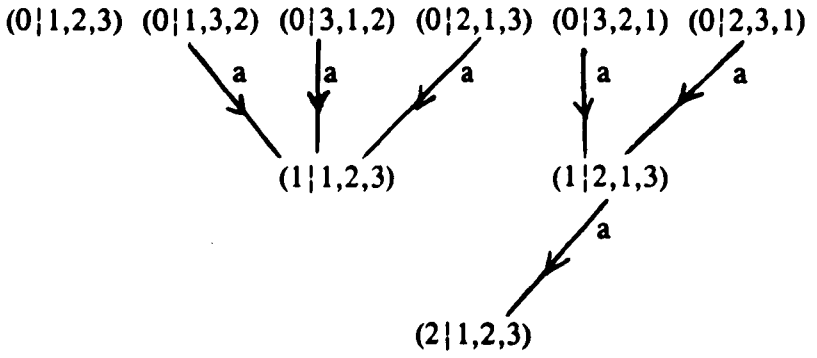
It is easily seen on the diagram that the object's procedural knowledge (to be defined in section 6) is to interchange x_i and x_{i+1} if $x_i > x_{i+1}$, for $i = 1,2$ from left to right.

EXAMPLE 3.3.2

Let O^2 denote the stand-alone object with nine local states such that:

$Q^2 = \{ (0|1,2,3), (0|1,3,2), (0|3,1,2), (0|2,3,1), (0|2,1,3), (0|3,2,1), (1|1,2,3), (1|2,1,3), (2|1,2,3) \}$,

$A^2 = \{ \epsilon, a \}$ with R^2_a defined by the following diagram:



The equivalence classes are: $\{(0|1,3,2),(0|3,1,2),(0|2,1,3),(1|2,1,3)\}$, $\{(0|3,2,1),(0|2,3,1)\}$ and $\{(0|1,2,3),(1|1,2,3),(2|1,2,3)\}$ and the relation $\bar{R}^2_a$ is defined by the following diagram:

$$\{(0|1,2,3), (1|1,2,3), (2|1,2,3)\}$$
$$|a$$
$$\uparrow$$
$$\{(0|1,3,2), (0|3,1,2), (0|2,1,3), (1|2,1,3)\}$$
$$|a$$
$$\uparrow$$
$$\{(0|3,2,1), (0|2,3,1)\}$$

The fact that the local states $(0|1,2,3)$, $(1|1,2,3)$ and $(2|1,2,3)$ are behaviorally indistinguishable, the same for $(0|2,1,3)$ and $(1|2,1,3)$ is an indication that the first digit in the local state specification is irrelevant to the overall behavior of the object, its

purpose being only procedural. It has a purely internal (or mental) significance [MH69]. It turns out as we will see, that this object is "behaviorally similar" to that of example 3.3.1.

In the following, we present our model of a DOBS computation referred to simply as system computation. The notion of similarity among system computations is also introduced.

We assume given for each object O^i in the DOBS an object instance (O^i,q^i_0).

3.4 SYSTEMS COMPUTATIONS

In section 2.5, we made some assumptions about the "general" liveness properties of distributed object-based systems. In particular, we assumed that each basic activity must be completed in unit time (with time ranging over the natural numbers). We also assumed that a message is received only if it has been sent earlier, and messages do not have to be received in the same order in which they have been sent. There is also the possibility that a message sent is never received.

Given an object instance (O^i,q^i_0) for each object O^i in the DOBS, we assume that at time 0, each object O^i is in state q^i_0.

An execution of the DOBS is a sequences of activities (including the null action) performed locally by each object in the system until some time $t \geq 0$. These activities are either the performance of a primitive action (internal event), a communication or external action initiation (send event) or the receipt of a message (receive event).

We denote by (i,a) the i-event (event on O^i) of performing the primitive action a, and for local state u, by $S(i,\bar{u},e,j)$ the i-event of sending the message $\bar{u}$ to O^j with respect to e and by $R(i,\bar{u},e,j)$ the i-event of receiving the message $\bar{u}$ sent by O^j with respect to e. Note that in our model a message is identified with the equivalence class of the state from which it is sent. For local states s,t of O^i, if we denote by:

$s \xrightarrow{(i,a)} t$ iff $(s,t) \in R^i_a$

$s \xrightarrow{S(i,\bar{u},e,j)} s$ iff $s \in \bar{u}$ and $(s,w,v) \in S^i_e$ for some $w,v \in Q^j$ and

$s \xrightarrow{R(i,\bar{u},e,j)} t$ iff $(u',s,t) \in S^i_e$ for some $u' \in \bar{u}$.

then a finite sequence of i-events $x^i = x^i_1 x^i_2 \ldots x^i_n$ is an i-computation iff there are local states $s_0 = q^i_0$, s_1 ,..., s_n such that $s_0 \ \underline{X^i_1}\ s_1, \ \ldots\ , s_{n-1} \ \underline{X_n}\ s_n$.

If $x^i = x^i_1 x^i_2 \ldots x^i_n$ is an i-computation, then we denote by $T^i(x^i)$ the set of local states $\{ t \mid q^i_0 \ \underline{X^i_1}\ s_1 ,\ldots , s_{n-1} \ \underline{X_n}\ t$, for some local states $s_1, \ldots, s_{n-1} \}$.
Note that this definition does not depend on whether message duplication is allowed or not (in the case message duplication is not allowed , x^i will be required to satisfy that property). This constrasts with the approach in [HF89] which allows a message to be duplicated by assuming that it is not removed from the buffer when it is received.

Our model of system computations is the same as that of [CM86] or [L89]. That means the linear interleaving model. If z is a finite sequence of events such that for each object instance (O^i,q^i_0) there is at least one i-event in z (z may also be required to satisfy the "bounded buffer" property or not, depending on whether the system has a bounded buffer or not) , then the i-projection z^i of z is the subsequence of z consisting of all i-events in z.

DEFINITION 3.4.1

z is a system computation of the DOBS if:

i) For each $i \geq 1$, z^i is an i-computation

ii) For every receive event $R(i,\bar{u},e,j)$ in z , there is a distinct send event $S(j,\bar{u},e,i)$ which occurs earlier in z.

If z is a system computation, then for each $i \geq 1$, $T^i(z) = T^i(z^i)$. The set of all system computations of the DOBS O will be denoted by $SC(O , q^1_0 , q^2_0 ,\ldots, q^n_0)$.

NOTATION: For sequences of events x , y we denote by xy the concatenation of x and y; and the fact that x is a prefix of y is written $x \leq y$. x is a proper prefix of y is written $x < y$.

Observe that system computations are prefix-closed. I.e. if y is a system computation and $x \leq y$ then x is a system computation.

3.5 COMPUTATIONS ISOMORPHISM

Intuitively, two system computations are isomorphic for an object instance if it cannot distinguish the result of the execution of one from that of the other. Formally, let x, y denote two system computations. We define the relation $x \sim_i y$ as follows:

$x \sim_i y$ iff

i) $\forall s \in T^i(x)$, $\exists t \in T^i(y)$ such that $s \sim_{Q^i} t$

ii) $\forall w \in T^i(y)$, $\exists u \in T^i(x)$ such that $w \sim_{Q^i} u$

PROPOSITION 3.5.1

The relation $\sim_i$ is an equivalence relation on $SC(O, q^1_0, q^2_0, \ldots, q^n_0)$.

PROOF: Let $\bar{T}^i(y) = \{ \bar{s} \mid s \in T^i(y) \}$. $x \sim_i y$ iff $\bar{T}^i(x) = \bar{T}^i(y)$.

DEFINITION 3.5.1

System computations x and y are isomorphic with respect to (O^i, q^i_0) (i-isomorphic) if $x \sim_i y$. They are isomorphic if they are i-isomorphic for each $i \geq 1$.

PROPOSITION 3.5.2: (Computation Extensions)

a) x is a system computation and y is a sequence of events such that xy is a system computation.

i) If there is no receive i-event in y, then xy^i is a system computation and $xy^i \sim_i xy$

ii) If every receive i-event in y has its corresponding send event in x, then xy^i is a system computation and $xy^i \sim_i xy$

b) x is a system computation and y^i is a sequence of i-events.

If xy^i is a system computation then $\forall j \neq i$, $xy^i \sim_j x$

c) x, z are system computations such that $x \sim_i z$ and y^i is a sequence of i-events not containing a receive event.

i) xy^i is a system computation iff zy^i is.

ii) If xy^i (respectively zy^i) is a system computation, then $xy^i \sim_i zy^i$

PROOF

a) $\forall j \neq i$, $(xy^i)^j = x^j$ and is a j-computation since x is a computation.

Also, $(xy^i)^i = x^i y^i = (xy)^i$. Therefore, if there is no receive i-event in y or if every receive i-event in y has its corresponding send event in x , then xy^i is a system computation (by definition) and $T^i(xy^i) = T^i(xy)$. Hence $xy^i \sim_i xy$.

b) $\forall j \neq i$, $(xy^i)^j = x^j$. Therefore, if xy^i is a system computation,

then $\forall j \neq i$, $T^j(xy^i) = T^j(x)$. Hence, $\forall j \neq i$, $xy^i \sim_j x$.

c) Let $y^i = y^i_1 y^i_2 \ldots y^i_n$ be the specification of y^i as a sequence of i-events. Since there is no receive event in y^i , xy^i is a system computation iff $\exists s_0 \in T^i(x)$ and

$s_1, s_2, \ldots s_n$ s.t. $s_0 \underline{y^i_1} s_1$, $s_1 \underline{y^i_2} s_2$, ... , $s_{n-1} \underline{y^i_n} s_n$.

But since $x \sim_i z$, $\exists t_0 \in T^i(z)$ s.t. $t_0 \sim_{Q^i} s_0$. And by definition of $\sim_{Q^i}$, there exist

$t_1 \sim_{Q^i} s_1$, $t_2 \sim_{Q^i} s_2$, ... , $t_n \sim_{Q^i} s_n$ s.t. $t_0 \underline{y^i_1} t_1$, $t_1 \underline{y^i_2} t_2$, ... , $t_{n-1} \underline{y^i_n} t_n$.

Hence, zy^i is a system computation and $\forall s \in T^i(xy^i)$, $\exists t \in T^i(zy^i)$ s.t. $s \sim_{Q^i} t$.

By symmetry of $\sim_i$, we also prove in the same way that if zy^i is a system computation, then xy^i is a system computation and $\forall t \in T^i(zy^i)$, $\exists s \in T^i(xy^i)$ s.t. $s \sim_{Q^i} t$. Therefore, if xy^i (respectively zy^i) is a system computation, then $xy^i \sim_i zy^i$.

4. BEHAVIORAL SIMILARITY - CLASS - INHERITANCE

In this section, we provide the mechanisms to formally characterize behaviorally similar objects, object classes and inheritance. These mechanisms are behavioral, and are similar to those suggested by O. Nierstrasz and M. Papathomas [NP90] or P. America and F. Van der Linden [AV90].

4.1 BEHAVIORAL SIMILARITY , OBJECT TYPE AND OBJECT CLASS

In section 2.7, we introduced the notion of behaviorally similar entities as corresponding to the situation in which the difference of their task-domains is irrelevant to their behavior specifications or protocols. In other words, there is a correspondance between their local states that is compatible with their behavior specifications, and also preserves behaviorally indistinguishable local states.

It turns out that this notion is very similar to that of O. Nierstrasz and M. Papathomas [NP90] or P. America and F. Van der Linden [AV90] object type. In [AV90], two objects are said to belong to the same type if "they share the same externally observable behavior". They also pointed out the need to differentiate the notion of object type from that of object class, observing that "a class describes how its instances are built, and a type describes how its elements can be used". According to [AV90], an object class is a collection of objects that have the same internal structure, that is the same instance variables, the same body and the same methods. But, since we are not concerned with objects' implementation, we will say that two objects belong to the same class if they are interchangeable. That means, if one can be substituted for the other and vice versa.

If O^1 and O^2 are object systems, then a function $f : Q^1 \text{-------} > Q^2$ will be said to verify property (B) if :

i) $\forall a \in A$ and $s, t \in Q^1$, if $(s,t) \in R^1_a$ then $(f(s),f(t)) \in R^2_a$

ii) $\forall e \in E$ and $s, t \in Q^1$,

if $\exists w \in \underline{Q}^1$, $(w,s,t) \in S^1_e$ then $\exists w' \in \underline{Q}^2$, $(w',f(s),f(t)) \in S^2_e$

iii) $\forall e \in E$ and $s \in Q^1$,

$\exists w, v \in \underline{Q}^1$ s.t. $(s,w,v) \in S^1_{e^-}$ iff $\exists w',v' \in \underline{Q}^2$ s.t. $(f(s),w',u') \in S^2_{e^-}$

iv) $\forall s, t \in Q^1$, $s \sim_{Q^1} t$ iff $f(s) \sim_{Q^2} f(t)$

DEFINITION 4.1.1

Object systems O^1 and O^2 are behaviorally similar, denoted by $O^1 \approx O^2$, if there are functions $f^1 : Q^1 \text{-------} > Q^2$ and $f^2 : Q^2 \text{-------} > Q^1$ such that :

i) f^1 and f^2 verify property (B), and

ii) $\forall s \in Q^1$, $f^2(f^1(s)) \sim_{Q^1} s$ and $\forall u \in Q^2$, $f^1(f^2(u)) \sim_{Q^2} u$.

Note that if O^1 and O^2 are behaviorally similar, then $A^1 = A^2$ and $E^1 = E^2$.
By definition, the relation $\approx$ is reflexive and symmetric; it is transitive by function composition.

PROPOSITION 4.1.1

The behavioral similarity relation is an equivalence relation.

REMARKS:

1) The functions f^j, $j = 1,2$ will be said to form a <u>pair of behavioral similarity functions</u>.

2) For behaviorally similar object systems O^1 and O^2, the corresponding pair of behavioral similarity functions need not be unique.

EXAMPLE 4.1.1

The objects O^1 of example 3.3.1 and O^2 of example 3.3.2 are behaviorally similar:
Let's define $f^1 : Q^1 \text{-----} > Q^2$ and $f^2 : Q^2 \text{----} > Q^1$ as follows:
$f^1(1,2,3)) = (2|1,2,3)$; $f^1((1,3,2)) = f^1((3,1,2)) = f^1((2,1,3)) = (1|2,1,3)$;
$f^1((2,3,1)) = (0|2,3,1)$; $f^1((3,2,1)) = (0|3,2,1)$; and
$f^2((0|1,2,3)) = f^2((1|1,2,3)) = f^2((2|1,2,3)) = (1,2,3)$; $f^2((0|1,3,2)) = (1,3,2)$;
$f^2((0|3,1,2)) = f^2(0|2,1,3)) = (3,1,2)$; $f^2((1|2,1,3)) = (2,1,3)$;
$f^2((0|2,3,1)) = (2,3,1)$; and $f^2((0|3,2,1)) = (3,2,1)$.
It is easy to see that f^1 and f^2 verify properties i) and ii) of defintion 4.1.1

As argued above, an object type is an equivalence class under the relation $\approx$.

DEFINITION 4.1.2: Object Type

An object type is a complete set T of pairwise behaviorally similar objects.

It is easy to see that null objects are behaviorally similar.
It thus follows that null objects constitute the null object type.

THEOREM 4.1.1

If O^1 and O^2 are behaviorally similar objects, then there is a bijection
$\bar{f} : \bar{Q}^1 \text{---------} > \bar{Q}^2$, such that :

1) $\forall a \in A$, $(\bar{s},\bar{t}) \in \bar{R}^1_a$ iff $(\bar{f}(\bar{s}),\bar{f}(\bar{t})) \in \bar{R}^2_a$
2) $\forall e \in E$,
   a) $\exists w \in Q^1\_$, $(w,\bar{s},\bar{t}) \in \bar{S}^1_e$ iff $\exists w' \in Q^2\_$, $(w',\bar{f}(\bar{s}),\bar{f}(\bar{t})) \in \bar{S}^2_e$
   b) $\forall \bar{s} \in \bar{Q}^1$, $\exists w, u \in Q^1\_$ s.t. $(s,w,u) \in S^1_e$- iff
      $\exists w', u' \in Q^2\_$ s.t. $(s',w',u') \in S^2_e$- for some $s' \in \bar{f}(\bar{s})$

Note that the above bijection need not be unique. For example, if O is the passive and private object system s.t. $Q = \{s, t\}$, S_e- $= \emptyset$ for $e \neq e_0$, and
S_{e_0}- $= \{(s,u,v), (t,u,w)\}$ for some u, v and w in $Q\_$ s.t. $v \neq w$,

then there are the identity and the bijection f from $\bar{Q}$ to $\bar{Q}$ s.t. $f(\bar{s}) = \bar{t}$ and $f(\bar{t}) = \bar{s}$ that verify the above properties.

PROOF

Since O^1 and O^2 are behaviorally similar, there is a pair of behavioral similarity functions $f^1 : Q^1 \dashrightarrow Q^2$ and $f^2 : Q^2 \dashrightarrow Q^1$.

Let $\bar{f}^1 : \bar{Q}^1 \dashrightarrow \bar{Q}^2$ s.t. $\bar{f}^1(\bar{s}) = \overline{f^1(s)}$, and $\bar{f}^2 : \bar{Q}^2 \dashrightarrow \bar{Q}^1$ defined similarly.

To show that $\bar{f}^1$ and $\bar{f}^2$ are well defined, note that from property (B)iv), it follows that $t \in \bar{s}$ iff $\bar{f}^j(\bar{t}) = \bar{f}^j(\bar{s})$, $j = 1, 2$. Therefore, $\bar{f}^j$, $j = 1, 2$ are functions.

It also follows from definition 4.1.1ii) that $\bar{f}^1(\bar{f}^2(\bar{t})) = \bar{t}$ and $\bar{f}^2(\bar{f}^1(\bar{s})) = \bar{s}$.

Hence, $\bar{f}^1$ is a bijection with inverse $\bar{f}^2$.

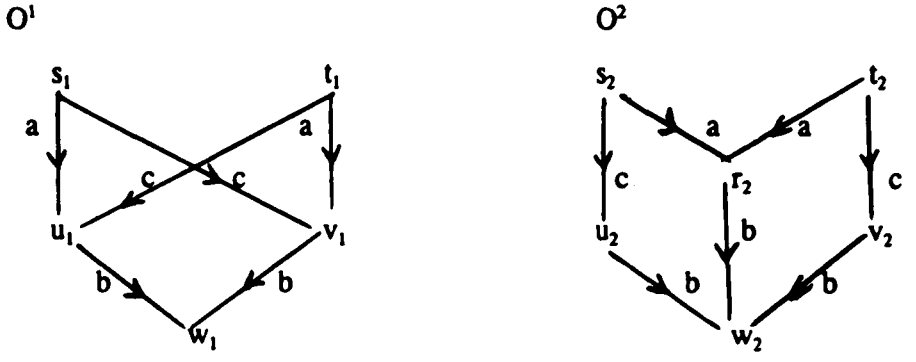
Property 1) (respectively property 2)a)) follows from the defintion of R_a , $a \in A$ and property (B)i) (respectively the definition of S_e , $e \in E$ and property (B)ii)).

Property 2)b) follows from property (B)iii) and the definition of $\bar{f}^1$.

REMARK: Objects O^1 and O^2 need not be behaviorally similar if there is a bijection $\bar{f}$ from $\bar{Q}^1$ to $\bar{Q}^2$ that satisfies the conditions of the above theorem.

For example, the following objects O^1 and O^2 are not behaviorally similar. However, there is one such bijection $\bar{f}$ from $\bar{Q}^1$ to $\bar{Q}^2$ defined as follows:

$\bar{f}(\{s_1, t_1\}) = \{s_2, t_2\}$; $\bar{f}(\{u_1, v_1\}) = \{u_2, r_2, v_2\}$; and $\bar{f}(\{w_1\}) = \{w_2\}$.



In the following, f is a function from Q^1 to Q^2 that satisfies property (B).

DEFINITION 4.1.3

i) f is an epimorphism from object system O^1 to object system O^2 if for each $s \in Q^1$ and $u \in Q^2$,

a) $\forall a \in A$, if $(f(s),u) \in R^2_a$, then $\exists t \in Q^1$ s.t. $f(t) = u$ and $(s,t) \in R^1_a$

b) $\forall e \in E$, if $(w,f(s),u) \in S^2_e$, for some w, then $\exists t \in Q^1$ s.t. $f(t) = u$ and $(w',s,t) \in S^1_e$ for some w'.

ii) Object system O^1 is said to be an epimorphic image of O^2, if there is an epimorphism from O^1 to O^2 that is surjective.

REMARK: object systems O^1 and O^2 need not be behaviorally similar if O^2 is an epimorphic image of O^1. For example, the following objects O^1 and O^2 are not behaviorally similar. However, O^2 is an epimorphic image of O^1.

DEFINITION 4.1.4: Object Class

Two object systems are said to belong to the same object class if they are epimorphic images of each other.

It is obvious that this relation is an equivalence relation on object systems. We will denote the equivalence class of object system O^i by $\bar{O}^i$. It is also easy to see that two object systems belong to the same object class only if they belong to the same object type. However, as we have noted above, the converse is not true.

A local state s of object system O^i will be said to be an isolated state if there is no communication initiation at s , and:

$\forall a \in A$, $a \neq \epsilon$ and $t \in Q^j$, $(s,t) \notin R^i_a$ and $(t,s) \notin R^i_a$

$\forall e \in E$, $t \in Q^j$ and $w \in \underline{Q^i}$, $(w,s,t) \notin S^i_e$ and $(w,t,s) \notin S^i_e$

We will denote by IS(i) the set of isolated states of object system O^i .

Note that if O^i is a null object, then all its local states are isolated states.

(I.e. $Q^i = IS(i)$).

In the following, the cardinality of a set S will be denoted by $|S|$.

4.2 INHERITANCE AND SUBCLASS

Our approach to inheritance corresponds to that of object specialization or incremental modification, and is closely related to that of O. Nierstrasz and M. Papathomas [NP90]. Object system O^j will be said to inherit from object system O^i if it can be viewed as an incremental modification of O^i , with which it shares some structure and some methods. In other words , O^j is viewed as deriving from O^i , possibly by addition of new local states, primitive actions, external properties, or possibilities to initiate communications.

Formally, let O^i , O^j be two object systems such that $A^i \subseteq A^j$, $E^i \subseteq E^j$ and $\forall e \in E$, $S^i_e\text{-} \neq \emptyset$ only if $S^j_e\text{-} \neq \emptyset$.

We denote by $O^j(i)$ the object system with set of local states $Q^j(i) = Q^j$ and set of states of the environment $\underline{Q^j}(i) = \underline{Q^j}$ such that:

$\forall a \in A$, $R^j_a(i) = R^j_a$ if $R^i_a \neq \emptyset$; otherwise $R^j_a(i) = \emptyset$ and

$\forall e \in E$, $S^j_e(i) = S^j_e$ if $S^i_e \neq \emptyset$; otherwise $S^j_e(i) = \emptyset$.

$S^j_e\text{-}(i) = S^j_e\text{-}$ if $S^i_e\text{-} \neq \emptyset$; otherwise $S^j_e\text{-}(i) = \emptyset$.

We denote by IS(j,i) the set of isolated states of $O^j(i)$.

DEFINITION 4.2.1: Inheritance

Object system O^j is said to inherit from object system O^i if :

a) $|IS(i)| \leq |IS(j,i)|$

b) There is a surjective function from $Q^j(i)-IS(j,i)$ to $Q^i-IS(i)$ that is an epimorphism from object system $O^j(i)$ to object system O^i.

Note that if O^i is an epimorphic image of O^j, then O^j inherits from O^i.

It also follows from the definition of object class that O^i and O^j inherit from each other if and only if they belong to the same object class (i.e. $\bar{O}^i = \bar{O}^j$).

Also, O^j inherits from O^i if and only if any member of the class of O^j inherits from every member of the class of O^i.

We then have the following definition of subclass relationship among object classes.

DEFINTION 4.2.2: Subclass

$\bar{O}^j$ is said to be a subclass of $\bar{O}^i$ if O^j inherits from O^i.

PROPOSITION 4.2.1

The subclass relationship is a partial order on object classes.

PROOF

It follows from the above remarks that the subclass relationship is reflexive and symmetric. It is transitive by composition of surjective functions.

5 REASONING ABOUT OBJECTS' KNOWLEDGE

There are two aspects of an object's knowledge in distributed environments: The first aspect that we refer to as dynamic knowledge corresponds to the knowledge obtained and transferred in distributed environments. That means what an object instance learns from other object instances in a system computation and otherwise would have not learned if there were no computations. In this respect, different instances of the same object need not learn the same things in a system computation. Also, what an object instance learns in a system computation may be different from what it learns in another system computation in which some of the object instances are replaced by others. It also corresponds to an object instance's awareness of its own knowledge, or properties of computations during a system computation and is closely related to the intuitive notion of learning by experience. On the other hand, the second aspect that we refer to as static/local knowledge is static and does not depend on object instances or system computations. It corresponds to facts about an object's task-domain situations that it should be said to know according to its behavior specification. In a multipartite game, this will correspond to the knowledge ascribed to each object based uniquely on its protocol.

The traditional approach to knowledge in distributed systems has mostly investigated the first aspect [CM86], [HF89], [Mos92], [Ne88], [Pa90] and [PR85], whereas little attention has been given to the second, despite the fact that its importance in constructing artificial intelligence and understanding natural intelligence has been noticed by J. McCarthy and P.J Hayes [MH69]: "... we regard the construction of intelligent machines as fact manipulators as being the best bet both for constructing artificial intelligence and understanding natural intelligence". In fact, in most formal treatments of knowledge in distributed environments, it is defined in terms of the first aspect (which by the way depends on system computations). In [CM86] it is referred to as "local predicate" and, if we denote by L_iA the fact that agent i learns (or is aware of) fact A (i.e. agent i knows fact A in [CM86]) and by K_iA the fact that agent i knows A (i.e. A is a local predicate of agent i in [CM86]) then according to [CM86],

a fact B is agent i's local predicate (i.e. B is of the form K_iA) if and only if the fact $L_iB \vee L_i\neg B$ is always true in all system computations. In other words, we always know whether B is true or not. Although this statement is true, it does not help us further to distinguish between "local knowledge" and knowledge gained during a computation since the latter also verifies this property. Namely, the formula $L_iL_iA \vee L_i\neg L_iA$ always holds, although the formula L_iA is not a local knowledge.

Our approach is to distinguish both aspects at the semantical level: We do "forget" what we learn or are aware of because what we learn or are aware of depends on the system dynamics. However, we do not forget what we know; but instead we change our knowledge state (or our view or perception of the world). Also, a system computation may be "consistent" with our local knowledge (in which case, we may use that knowledge) or not (in which case, we are not sure about it). Note that our notion of "sure" is different from that of [CM86]. In the following, we would like to present a formal treatment of this aspect of an object's knowledge. Further discussions on dynamic knowledge, its formal treatment and relationship to local knowledge are postponed until section 7.

It turns out that our formal system corresponds to a logic of knowledge and actions. Our approach is based upon the intuition that every intelligent protocol is governed by the agent's knowledge: Knowledge of facts about its task-domain situations, and knowledge of what to achieve (and how to achieve it) at any given local state. Very often, we hear people asking themselves: "What am I going to do now?". Note that, not only do we need to know what we can do in certain circumstances, but we also need to know what we will achieve by doing what we actually choose to do. In fact, in most situations in real life such as game playing or mathematical reasoning, this aspect of knowledge has been used to classify individuals according to their intelligence. It is also worth mentioning that it is also used in most tests of intelligence. For example, to find out if a child knows what a triangle is, (s)he is presented with many situations where (s)he has to recognize a triangle, and the conclusion is based on whether or not (s)he makes mistakes.

Since at any local state an entity's protocol is determined by its knowledge at that state, for an observer, an entity knows a fact A (denoted by KA) at a local state w if and only if A is true at all local states behaviorally indistinguishable to w.

This aspect of knowledge may be of great interest in Expert Systems and in particular in the area of Knowledge Elicitation and Formalization where experts' knowledge is derived from their behavior specifications. In this particular context, the "logical omniscience" property of the knower turns out to be a desired property since from the logical system, facts about an expert's knowledge can be derived even though the expert could have not derived them. The logical system may also be a useful tool in detecting inconsistency in experts' behaviors. An object's behavior will be said to be inconsistent with respect to its knowledge if different behaviors are observed at local states in which it has identical knowledge. This notion is formalized in section 6 and is similar to that referred to as "irrational behavior" in [MH69] where other relevant issues are discussed. This approach to knowledge is also apparent in most experimental sciences such as biology and medical sciences where knowledge about phenomena is derived from experiments. Note that the logical system may also be a useful tool for the knowledge-based analysis of systems.

In the following, we introduce the syntax and the semantics of a language for talking about objects' local/static knowledge. In section 5.2, we propose a deductive system for the valid formulas and also prove its soundness. We prove its completeness and decidability in section 5.3, and in section 5.4, we prove some properties of "frames" related to behaviorally similar and epimorphic objects. The semantics of objects is introduced in section 6 where we also provide a knowledge-based characterization of behavioral similarity, inheritance, and rational behavior.

5.1 LANGUAGE AND ITS SEMANTICS

To introduce the syntax of our language, we need a countable set Φ of atomic formulas and the alphabets A and E of primitive actions and external properties respectively.

For $a \in A$, we denote by the same symbol a the basic activity of performing the primitive action a, and for $e \in E$, we denote by (:e) the basic activity of executing the external action initiated with respect to external property e. Note that receiving a message sent with respect to e corresponds to performing some basic activity represented by (:e).

The set ACT of activities is defined by induction as follows:

i) $\forall a \in A$ and $e \in E$, $a \in ACT$ and $(:e) \in ACT$

ii) if $\alpha, \beta \in ACT$, then so are: $\alpha\beta$ (concatenation), $\alpha \cup \beta$ (non-deterministic selection), α^* (finite repetition), with the following meaning:

$\alpha\beta$ perform α and then β

$\alpha \cup \beta$ perform either α or β non-deterministically

α^* perform α some finite number (≥ 0) of times

We now define the language $\mathcal{L}(\Phi, A, E)$ of formulas as follows:

$\top \in \mathcal{L}$ and

if $A \in \Phi$ and $e \in E$, then A and $(\rightarrow e) \in \mathcal{L}$

if $A, B \in \mathcal{L}$ then so are $\neg A$, $A \wedge B$, KA

if $\alpha \in ACT$ and $A \in \mathcal{L}$ then $<\alpha>A \in \mathcal{L}$

K is the knowledge operator (KA is read: the object knows A) and $<\alpha>A$ says that there is an execution of α that brings about A. For $e \in E$, the formula $(\rightarrow e)$ states that it is possible to initiate a communication (or an action) with respect to external property e. We also use the following syntactic abbreviations where A, B are formulas and $\alpha \in ACT$: $\bot := \neg\top$; $A \vee B := \neg(\neg A \wedge \neg B)$; $A \Rightarrow B := \neg A \vee B$; $A \Leftrightarrow B := (A \Rightarrow B) \wedge (B \Rightarrow A)$ and $[\alpha]A := \neg <\alpha> \neg A$.

The formula $[\alpha]A$ says that every execution of α brings about A.

Given an object system O, an $\mathcal{L}$-frame is a structure $F = (O, \sim_Q)$, and an interpretation of the atomic formulas is a map $V : \Phi \dashrightarrow \mathcal{P}(Q)$. We do not assume that if $t \sim_Q s$ and $s \in V(p)$, then $t \in V(p)$.
A model on the frame F is a pair $M = (F, V)$, where V is an interpretation of the atomic formulas.

Given a frame F and $\alpha \in ACT$, we define the relation $\underline{\alpha} \subseteq Q \times Q$ by induction on the structure of α as follows:

$\forall a \in A, e \in E$, $w \underline{a} w'$ iff $(w,w') \in R_a$ and

$w \underline{(:e)} w'$ iff $\exists s \in Q$ s.t. $(s,w,w') \in S_e$.

$\forall \alpha, \beta \in ACT$,

$w \underline{\alpha\beta} w'$ iff $\exists u$ s.t. $w \underline{\alpha} u$ and $u \underline{\beta} w'$

$w \underline{\alpha \cup \beta} w'$ iff $w \underline{\alpha} w'$ or $w \underline{\beta} w'$

$w \underline{\alpha^*} w'$ iff $\exists u_0 = w, u_1, \ldots, u_n = w'$ $(n \geq 0)$ s.t.

$u_0 \underline{\alpha} u_1, u_1 \underline{\alpha} u_2, \ldots, u_{n-1} \underline{\alpha} u_n$

PROPOSITION 5.1.1

$\forall \alpha \in ACT$ and $s,t \in Q$, if $s \underline{\alpha} t$, then $\forall s' \in \bar{s}$, $\exists t' \in \bar{t}$ s.t. $s' \underline{\alpha} t'$.

PROOF: By induction on the structure of α.

The initial case $\alpha = a \in A$ or $\alpha = (:e)$, $e \in E$ follows from proposition 3.3.3.a)

If we assume the hypothesis true for α and β, then:

i) $s \underline{\alpha\beta} t$ iff $\exists u$ s.t. $s \underline{\alpha} u$ and $u \underline{\beta} t$ implies $\forall s' \in \bar{s}$, $\exists u' \in \bar{u}$ s.t. $s' \underline{\alpha} u'$ and $u \underline{\beta} t$; which implies $\forall s' \in \bar{s}$, $\exists u' \in \bar{u}$ and $t' \in \bar{t}$ s.t. $s' \underline{\alpha} u'$ and $u' \underline{\beta} t'$.
Hence $\forall s' \in \bar{s}$, $\exists t' \in \bar{t}$ s.t. $s' \underline{\alpha\beta} t'$.

ii) $s \underline{\alpha \cup \beta} t$ iff $s \underline{\alpha} t$ or $s \underline{\beta} t$ implies $\forall s' \in \bar{s}$, $\exists t' \in \bar{t}$ s.t. $s' \underline{\alpha} t'$ or $\forall s' \in \bar{s}$, $\exists t'' \in \bar{t}$ s.t. $s' \underline{\beta} t''$. Therefore $\forall s' \in \bar{s}$, $\exists t' \in \bar{t}$ s.t. $s \underline{\alpha \cup \beta} t'$ or $\forall s' \in \bar{s}$, $\exists t'' \in \bar{t}$ s.t. $s' \underline{\alpha \cup \beta} t''$. Hence, $\forall s' \in \bar{s}$, $\exists t' \in \bar{t}$ s.t. $s' \underline{\alpha \cup \beta} t'$.

iii) $s \underline{\alpha^*} t$ iff $\exists s_0=s, s_1, \ldots, s_n=t, n \geq 0$ s.t. $s_0 \underline{\alpha} s_1, \ldots, s_{n-1} \underline{\alpha} s_n$.
Which implies $\forall s' \in \bar{s}, \exists s_1, t_1 \in \bar{s}_1, s_2, \ldots, s_n=t$ s.t. $s' \underline{\alpha} t_1, s_1 \underline{\alpha} s_2, \ldots, s_{n-1} \underline{\alpha} s_n$.
By succesive application of the induction hypothesis on $t_i \in \bar{s}_i$ and $s_i \underline{\alpha} s_{i+1}$, $i \leq n-1$, we obtain $t_1, t_2, \ldots, t_n \in \bar{t}$ s.t. $s' \underline{\alpha} t_1$, $t_1 \underline{\alpha} t_2, \ldots, t_{n-1} \underline{\alpha} t_n$.
Therefore, $\forall s' \in \bar{s}, \exists t'=t_n \in \bar{t}$ s.t. $s' \underline{\alpha^*} t'$.

Given a model M and $w \in Q$, we define the satisfaction relation $M,w \models A$ by induction on the structure of $A \in \mathcal{L}$ as follows:

$M,w \models \top$ and

if $A \in \Phi$, then $M,w \models A$ iff $w \in V(A)$

$M,w \models (\rightarrow e)$ iff $\exists s,t \in Q$ s.t. $(w,s,t) \in S_e$-

$M,w \models \neg A$ iff not $M,w \models A$

$M,w \models A \wedge B$ iff $M,w \models A$ and $M,w \models B$

$M,w \models KA$ iff if $w' \sim_Q w$, then $M,w' \models A$

$M,w \models \langle \alpha \rangle A$ iff there is a u s.t. $w \underline{\alpha} u$ and $M,u \models A$

A formula $A \in \mathcal{L}$ is said to be true in model M, denoted by $M \models A$ if $\forall w \in Q$, $M,w \models A$. It is valid in frame F, denoted by $F \models A$ if it is true in all models M based on F. It is valid, denoted by $\models A$ if it is valid in all frames.

5.2 PROPOSITIONAL LOGIC OF KNOWLEDGE AND ACTIONS

We propose an axiomatization of the valid formulas, and also prove its soundness.

LEMMA 5.2.1

The following formula schemata are valid.

1. $<\epsilon>A \Leftrightarrow A$
2. $(\rightarrow e) \Leftrightarrow K(\rightarrow e)$
3. $KA \Rightarrow A$
4. $KA \wedge K(A \Rightarrow B) \Rightarrow KB$
5. $K\neg<\alpha>KA \vee K<\alpha>KA$
6. $K\neg[\alpha]KA \vee K[\alpha]KA$
7. $<\alpha>\perp \Leftrightarrow \perp$
8. $<\alpha\cup\beta>A \Leftrightarrow <\alpha>A \vee <\beta>A$
9. $<\alpha\beta>A \Leftrightarrow <\alpha><\beta>A$
10. $<\alpha>(A \vee B) \Leftrightarrow <\alpha>A \vee <\alpha>B$
11. $(A \vee <\alpha><\alpha^*>A) \Rightarrow <\alpha^*>A$
12. $<\alpha^*>A \Rightarrow (A \vee <\alpha^*>(\neg A \wedge <\alpha>A))$

Axioms 3 and 4 are axioms of the logic S5 of knowledge [G87], [GN87] and axioms 7 to 12 are axioms of the Propositional Dynamic Logic (PDL) of regular programs [G87], [KT89], [KP81] and their proofs are the same as for those logics.

Axioms 5 and 6 are the "abilities" axioms. They express the fact that an agent always knows whether it can achieve a goal or not. More precisely, an observer or system analyst can always find out whether an agent can achieve a goal or not. Note that a goal is represented as local knowledge.

We will provide the proofs for axioms 1, 2, 5 and 6.

PROOF:

1. $M,s \models \langle\epsilon\rangle A$ iff $M,s \models A$, since $s \underline{\epsilon}\ t$ iff $s=t$

2. $M,s \models (\rightarrow e)$ iff $M,t \models (\rightarrow e)$, $\forall t \sim_Q s$ (by definition of $\sim_Q$) iff $M,s \models K(\rightarrow e)$.

5. $M,s \not\models K\neg\langle\alpha\rangle KA$ iff $\exists t \sim_Q s$ s.t. $M,t \models \langle\alpha\rangle KA$ iff $\exists t \sim_Q s$ and u s.t. $t \underline{\alpha} u$ and $M,u \models KA$. Let v s.t. $v \sim_Q s$. By proposition 5.1.1 , $\exists w \sim_Q u$ s.t. $v \underline{\alpha} w$. Therefore, $\exists w$ s.t. $v \underline{\alpha} w$ and $M,w \models KA$. I.e. $M,v \models \langle\alpha\rangle KA$. Hence, $M,s \models K\langle\alpha\rangle KA$.

6. $M,s \not\models K\neg[\alpha]KA$ iff $\exists t \sim_Q s$ s.t. $M,t \models [\alpha]KA$ iff

$\exists t \sim_Q s$ s.t. if $t \underline{\alpha} u$ then $M,u \models KA$. Let v s.t. $v \sim_Q s$.

By proposition 5.1.1 , if $v \underline{\alpha} w$ then $\exists w' \sim_Q w$ s.t. $t \underline{\alpha} w'$.

Therefore if $v \underline{\alpha} w$ then $\exists w' \sim_Q w$ s.t. $M,w' \models KA$.

Hence, if $v \underline{\alpha} w$, then $M,w \models KA$. I.e. $M,v \models [\alpha]KA$. So, $M,s \models K[\alpha]KA$.

LEMMA 5.2.2

The following rules of inference are sound:

$\dfrac{A}{KA}$ Generalization of K; $\qquad \dfrac{A \Rightarrow B}{\langle\alpha\rangle A \Rightarrow \langle\alpha\rangle B}$ monotonicity of $\langle\alpha\rangle$

PROOF: The first is a rule of the logic S5 of knowledge [G87] and the second is a rule of PDL [KP81], and their proofs are the same as for those logics.

DEFINITION 5.2.1 : Propositional Logic of Knowledge and Actions (PLKA)

The Propositional Logic of Knowledge and Actions (PLKA) is the smallest set of formulas containing all tautologies, the formula schemata 1 to 12, and closed under the rules of generalization of K , monotonicity of $\langle\alpha\rangle$, and modus ponens.

We write $\vdash A$ if A is a theorem of PLKA. A formula A is <u>consistent</u> if not $\vdash \neg A$, and a set of formulas Γ is <u>consistent</u> if all finite conjunctions of elements of Γ are consistent.

REMARKS:

It follows from the above definitions that A is inconsistent iff $\vdash \neg A$ iff $\vdash A \Rightarrow \bot$ and $\{B_1, B_2, \ldots, B_n\} \cup \{A\}$ is inconsistent iff $\vdash B_1 \wedge \ldots \wedge B_n \wedge A \Rightarrow \bot$ iff $\vdash (B_1 \wedge \ldots \wedge B_n) \Rightarrow \neg A$ iff $\vdash \neg B_1 \vee \ldots \vee \neg B_n \vee \neg A$.

THEOREM 5.2.1

The deductive system for PLKA is sound. I.e. if $\vdash A$ then $\models A$.

PROOF The proof follows from lemmas 5.2.1 and 5.2.2.

A formula A is a tautological consequence of the formulas $A_1, A_2, \ldots, A_n$, $n \geq 1$ if the formula $(A_1 \wedge A_2 \wedge \ldots \wedge A_n) \Rightarrow A$ is a tautology. Note that PLKA is closed under tautological consequence. I.e. if $\vdash A_1, \vdash A_2, \ldots, \vdash A_n$ and $(A_1 \wedge A_2 \wedge \ldots \wedge A_n) \Rightarrow A$ is a tautology , then $\vdash A$. We will make several uses of this closure property.

THEOREM 5.2.2

PLKA is closed under the following rules of inference:

$$\frac{A \Rightarrow B}{KA \Rightarrow KB} \qquad \frac{A \Rightarrow B}{[\alpha]A \Rightarrow [\alpha]B}$$ (monotonicity of K and $[\alpha]$)

$$\frac{A}{[\alpha]A}$$ (generalization of $[\alpha]$) and $$\frac{A \Rightarrow \langle\alpha\rangle B,\ B \Rightarrow \langle\beta\rangle C}{A \Rightarrow \langle\alpha\beta\rangle C}$$

PROOF

i) If $\vdash A \Rightarrow B$ then $\vdash K(A \Rightarrow B)$ by generalization of K.

But , $\vdash K(A \Rightarrow B) \Rightarrow (KA \Rightarrow KB)$ by axiom 4 and tautological consequence.

Therefore , $\vdash KA \Rightarrow KB$ by modus ponens.

ii) $\vdash A \Rightarrow B$ iff $\vdash \neg B \Rightarrow \neg A$ by tautological consequence.

Hence, $\vdash \langle\alpha\rangle \neg B \Rightarrow \langle\alpha\rangle \neg A$ by monotonicity of $\langle\alpha\rangle$.

Therefore, $\vdash [\alpha]A \Rightarrow [\alpha]B$ by tautological consequence.

iii) $\vdash A$ iff $\vdash \neg A \Rightarrow \bot$ by tautological consequence.

Hence , $\vdash <\alpha> \neg A \Rightarrow <\alpha> \bot$ by monotonicity of $<\alpha>$.

Therefore, $\vdash [\alpha]A$ by axiom 7 and tautological consequence.

iv) If $\vdash A \Rightarrow <\alpha>B$ and $\vdash B \Rightarrow <\beta>C$

then , $\vdash A \Rightarrow <\alpha>B$ and $\vdash <\alpha>B \Rightarrow <\alpha><\beta>C$ by monotonicity of $<\alpha>$.

Therefore , $\vdash A \Rightarrow <\alpha\beta>C$ by axiom 9 and tautological consequence.

THEOREM 5.2.3

The following formula schemata are theorems of PLKA.

a) $[\epsilon]A \Leftrightarrow A$

b) $K(A \wedge B) \Leftrightarrow KA \wedge KB$

c) $KA \vee KB \Rightarrow K(A \vee B)$

d) $<\alpha>KA \Rightarrow K<\alpha>KA$

e) $<\alpha> \neg KA \Rightarrow K<\alpha> \neg KA$

f) $[\alpha]KA \Rightarrow K[\alpha]KA$

g) $[\alpha] \neg KA \Rightarrow K[\alpha] \neg KA$

h) $KA \Rightarrow KKA$

i) $\neg KA \Rightarrow K \neg KA$

j) $[\alpha]A \wedge [\alpha](A \Rightarrow B) \Rightarrow [\alpha]B$

k) $[\alpha]A \vee [\alpha]B \Rightarrow [\alpha](A \vee B)$

l) $<\alpha>(A \wedge B) \Rightarrow <\alpha>A \wedge <\alpha>B$

m) $[\alpha]T \Leftrightarrow T$

n) $[\alpha \cup \beta]A \Leftrightarrow [\alpha]A \wedge [\beta]A$

o) $[\alpha\beta]A \Leftrightarrow [\alpha][\beta]A$

p) $[\alpha](A \wedge B) \Leftrightarrow [\alpha]A \wedge [\alpha]B$

q) $[\alpha^*]A \Leftrightarrow (A \wedge [\alpha][\alpha^*]A)$

r) $A \wedge [\alpha^*](\neg A \vee [\alpha]A) \Leftrightarrow [\alpha^*]A$

PROOF

a) From axiom 1, $\vdash <\epsilon> \neg A \Leftrightarrow \neg A$. Hence $\vdash [\epsilon]A \Leftrightarrow A$ by tautological consequence.

b) $\vdash A \wedge B \Rightarrow A$ and $\vdash A \wedge B \Rightarrow B$ (tautologies). It then follows from the monotonocity of K and tautological consequence that $\vdash K(A \wedge B) \Rightarrow KA \wedge KB$.

Also, $\vdash A \Rightarrow (B \Rightarrow A \wedge B)$ (tautology). Therefore $\vdash KA \wedge KB \Rightarrow K(A \wedge B)$ by monotonicity of K and tautological consequence.

c) $\vdash A \Rightarrow A \vee B$ and $\vdash B \Rightarrow A \vee B$ (tautologies). Therefore,

$\vdash KA \vee KB \Rightarrow K(A \vee B)$ by monotonicity of K and tautological consequence.

d) It follows from axiom 3 and tautological consequence that

$\vdash \neg[\alpha]\neg KA \Rightarrow \neg K[\alpha]\neg KA$ and from axiom 5 and tautological consequence we have $\vdash <\alpha> KA \Rightarrow K<\alpha> KA$.

e) It follows from axiom 3 and tautological consequence that $\vdash \neg[\alpha]KA \Rightarrow \neg K[\alpha]KA$ and from axiom 6 and tautological consequence we get $\vdash <\alpha> \neg KA \Rightarrow K<\alpha> \neg KA$.

f) and g) are proved similarly to d) and e) respectively, by using axiom 6 for f) and axiom 5 for g).

h) follows from f) and a) and i) follows from g) and a) by taking $\alpha = \epsilon$.

j) and k) are proved in the same way as b) and c) respectively, by using the monotonicity of $[\alpha]$. l) is dual of k) and m) to p) are duals of axioms 7 to 10 respectively.

q) and r): $\vdash [\alpha^*]A \Rightarrow (A \wedge [\alpha][\alpha^*]A)$ and $\vdash A \wedge [\alpha^*](\neg A \vee [\alpha]A) \Rightarrow [\alpha^*]A$ (duals of axioms 11 and 12 respectively). From o) and tautological consequence we have $\vdash [\alpha^*][\alpha]A \Leftrightarrow [\alpha][\alpha^*]A$. Also, $\vdash [\alpha]A \Rightarrow \neg A \vee [\alpha]A$ (tautology).

Then $\vdash [\alpha^*][\alpha]A \Rightarrow [\alpha^*](\neg A \vee [\alpha]A)$ by monotonicity of $[\alpha^*]$.

Hence , $\vdash A \wedge [\alpha^*][\alpha]A \Rightarrow A \wedge [\alpha^*](\neg A \vee [\alpha]A)$ by tautological consequence.

Therefore, $\vdash A \wedge [\alpha][\alpha^*]A \Rightarrow [\alpha^*]A$ and $\vdash [\alpha^*]A \Rightarrow A \wedge [\alpha^*](\neg A \vee [\alpha]A)$ by tautological consequence.

REMARK Notice that tautologies, axioms 3 and 4, and theorems h) and i) are the axioms of the logic S5 of knowledge, which is closed under modus ponens and generalization of K rules of inference. Also, tautologies, axioms 7 to 12 are the axioms of PDL of regular programs, which is also closed under modus ponens and monotonicity of $<\alpha>$ rules of inference. It thus follows that PLKA is an extension of both the logic S5 of knowledge, and PDL of regular programs.

5.3 COMPLETENESS AND DECIDABILITY

PLKA is complete iff $\models A$ implies $\vdash A$. In other words, if $\not\vdash \neg A$ then $\not\models \neg A$. Which is equivalent to saying that every consistent formula is satisfiable. It is then enough to show that every consistent formula has a model. We do so by constructing a finite model for every consistent formula and thus, proving at the same time the decidability of PLKA.

DEFINITION 5.3.1

A PLKA-maximal set is a consistent set of formulas Γ s.t. for any formula A, either $A \in \Gamma$ or $\neg A \in \Gamma$.

The set of PLKA-maximal sets will be denoted by MS(PLKA). Note that PLKA-maximal sets are not closed under the above rules of inference.
The proof of the following proposition may be found in [G87].

PROPOSITION 5.3.1: Lindenbaum's lemma

Every consistent set of formulas is contained in a PLKA-maximal set.

PROPOSITION 5.3.2

If Γ is a PLKA-maximal set, then:

a) For any formula A, $A \in \Gamma$ iff $\neg A \notin \Gamma$.

b) PLKA $\subseteq \Gamma$.

c) If $A \in \Gamma$ and $(A \Rightarrow B) \in \Gamma$ then $B \in \Gamma$.

d) $A \in \Gamma$ and $B \in \Gamma$ iff $(A \wedge B) \in \Gamma$.

e) $A \in \Gamma$ or $B \in \Gamma$ iff $(A \vee B) \in \Gamma$.

f) If $A \in \Gamma$ and $\vdash A \Rightarrow B$ then $B \in \Gamma$.

PROOF

a) $A \in \Gamma$ or $\neg A \in \Gamma$ and $\vdash A \wedge \neg A \Rightarrow \bot$.

Then by consistency of Γ, $A \in \Gamma$ iff $\neg A \notin \Gamma$.

b) $\vdash A$ iff $\vdash \neg A \Rightarrow \bot$. Then by consistency of Γ, $\neg A \notin \Gamma$ and by a), $A \in \Gamma$.

c) $\vdash A \wedge (A \Rightarrow B) \wedge \neg B \Rightarrow \bot$ (tautology). Therefore, $A, (A \Rightarrow B) \in \Gamma$ implies $\neg B \notin \Gamma$ by consistency of Γ; and by a), $B \in \Gamma$.

d) $A \Rightarrow (B \Rightarrow A \wedge B) \in \Gamma$ by b).

Therefore, $A, B \in \Gamma$ implies $(A \wedge B) \in \Gamma$ by c).

On the other hand, $(A \wedge B \Rightarrow A) \in \Gamma$ and $(A \wedge B \Rightarrow B) \in \Gamma$ by b).

Therefore, $(A \wedge B) \in \Gamma$ implies $A, B \in \Gamma$ by c).

e) is dual of d) and f) follows from b) and c).

PROPOSITION 5.3.3

s, t are PLKA-maximal sets and $\alpha \in ACT$.

a) $\langle\alpha\rangle A \in s$ iff there is a PLKA-maximal set u s.t.

$$\{B \mid [\alpha]B \in s\} \cup \{A\} \subseteq u$$

b) $\{A \mid [\alpha]A \in s\} \subseteq t$ iff $\{\langle\alpha\rangle B \mid B \in t\} \subseteq s$

PROOF

a) If $\{B \mid [\alpha]B \in s\}\cup\{A\}$ is inconsistent, then there exist $B_1, \ldots, B_n$ s.t $[\alpha]B_i \in s$, $i \leq n$ and $\vdash B_1 \wedge \ldots \wedge B_n \Rightarrow \neg A$. Hence, $\vdash [\alpha]B_1 \wedge \ldots \wedge [\alpha]B_n \Rightarrow [\alpha]\neg A$ by monotonicity of $[\alpha]$, theorem 5.2.3p) and tautological consequence. By propositions 5.3.2d) and 5.3.2f), we obtain $\neg<\alpha>A \in s$. Which is impossible by proposition 5.3.2a). Hence $\{B \mid [\alpha]B \in s\}\cup\{A\}$ is consistent and by Lindenbaum's lemma, there is a PLKA-maximal set u s.t. $\{B \mid [\alpha]B \in s\}\cup\{A\} \subseteq u$. On the other hand, if $\{B \mid [\alpha]B \in s\}\cup\{A\} \subseteq u$, then by proposition 5.3.2a), $<\alpha>A \in s$.

b) If $\{A \mid [\alpha]A \in s\} \subseteq t$ and $B \in t$, then $<\alpha>B \in s$ by part a). On the other hand, if $\{<\alpha>B \mid B \in t\} \subseteq s$ and $[\alpha]A \in s$, then by proposition 5.3.2a), $A \in t$.

PROPOSITION 5.3.4

s , t are PLKA-maximal sets and $\alpha, \beta \in ACT$.

a) If $\{A \mid [\alpha\beta]A \in s\} \subseteq t$, then there is a PLKA-maximal set u s.t.
$\{A \mid [\alpha]A \in s\} \subseteq u$ and $\{B \mid [\beta]B \in u\} \subseteq t$

b) If $\{A \mid [\alpha\cup\beta]A \in s\} \subseteq t$
then $\{A \mid [\alpha]A \in s\} \subseteq t$ or $\{B \mid [\beta]B \in s\} \subseteq t$

PROOF

a) If $\{A \mid [\alpha\beta]A \in s\} \subseteq t$ and $\{A \mid [\alpha]A \in s\}\cup\{<\beta>B \mid B \in t\}$ is inconsistent, then there exist $A_1, \ldots, A_n$, $[\alpha]A_i \in s$ $\forall i \leq n$, and $B_1, \ldots, B_m$, $B_j \in t$ $\forall j \leq m$ s.t. $\vdash A_1 \wedge \ldots \wedge A_n \Rightarrow \neg(<\beta>B_1 \wedge \ldots \wedge <\beta>B_m)$. And from theorem 5.2.3l) and tautological consequence it follows that
$\vdash A_1 \wedge \ldots \wedge A_n \Rightarrow \neg<\beta>(B_1 \wedge \ldots \wedge B_m)$.
Hence, $\vdash [\alpha]A_1 \wedge \ldots \wedge [\alpha]A_n \Rightarrow [\alpha]\neg<\beta>(B_1 \wedge \ldots \wedge B_m)$ by monotonicity of $[\alpha]$, theorem 5.2.3p) and tautological consequence.
Therefore, $[\alpha]\neg<\beta>(B_1 \wedge \ldots \wedge B_m) \in s$ by propositions 5.3.2d) and 5.3.2f). And from theorem 5.2.3o) and proposition 5.3.2f) it follows that $[\alpha\beta]\neg(B_1 \wedge \ldots \wedge B_m) \in s$.
Hence, $\neg(B_1 \wedge \ldots \wedge B_m) \in t$.

Which is impossible by proposition 5.3.2a) since $B_1 \wedge \ldots \wedge B_n \in t$ by proposition 5.3.2d). Therefore, $\{A \mid [\alpha]A \in s\} \cup \{<\beta>B \mid B \in t\}$ is consistent and there is a PLKA-maximal set u s.t. $\{A \mid [\alpha]A \in s\} \cup \{<\beta>B \mid B \in t\} \subseteq u$ by Lindenbaum's lemma.

From proposition 5.3.3b) it follows that $\{A \mid [\alpha]A \in s\} \subseteq u$ and $\{B \mid [\beta]B \in u\} \subseteq t$.

b) If $\{A \mid [\alpha]A \in s\} \not\subseteq t$ and $\{B \mid [\beta]B \in s\} \not\subseteq t$ then there are formulas $[\alpha]A \in s$ and $[\beta]B \in s$ s.t. $\neg A \in t$ and $\neg B \in t$ by proposition 5.3.2.a).

But $[\alpha]A \vee [\alpha]B \in s$ and $[\beta]A \vee [\beta]B \in s$ by proposition 5.3.2e).

Therefore, $[\alpha](A \vee B) \in s$ and $[\beta](A \vee B) \in s$ by theorem 5.2.3k) and proposition 5.3.2f). Hence, $[\alpha](A \vee B) \wedge [\beta](A \vee B) \in s$ by proposition 5.3.2d) ; from theorem 5.2.3n) and proposition 5.3.2f) we obtain $[\alpha \cup \beta](A \vee B) \in s$.

Hence, $A \vee B \in t$ by hypothesis. But this is impossible by proposition 5.3.2e).

Therefore, $\{A \mid [\alpha]A \in s\} \subseteq t$ or $\{B \mid [\beta]B \in s\} \subseteq t$.

We define the class of <u>K-formulas</u> Kf by induction as follows:

For any formula A , $KA \in Kf$.

If $A, B \in Kf$ and $\alpha \in ACT$, then $\neg A$, $<\alpha>A$ and $A \wedge B$ belong to Kf.

Note that if $A, B \in Kf$ and $\alpha \in ACT$ then by duality, $[\alpha]A$ and $A \vee B$ also belong to Kf.

LEMMA 5.3.1

If B is a K-formula , then $\vdash B \Rightarrow KB$

PROOF By induction on the structure of B

i) The initial case $B = KA$ follows from theorem 5.2.3 h)

ii) Let $B = \neg C$ s.t. $\vdash C \Rightarrow KC$. Then $\vdash K\neg KC \Rightarrow K\neg C$ by tautological consequence and monotonicity of K. Hence, $\vdash \neg KC \Rightarrow K\neg C$ by theorem 5.2.3 i) and tautological consequence. Therefore, $\vdash \neg C \Rightarrow K\neg C$ by axiom 3 and tautological consequence.

iii) Let $B = <\alpha>C$ s.t. $\vdash C \Rightarrow KC$. Then $\vdash <\alpha>C \Rightarrow K<\alpha>KC$ by monotonicity of $<\alpha>$, theorem 5.2.3d) and tautological consequence.
But $\vdash K<\alpha>KC \Rightarrow K<\alpha>C$ by axiom 3 , monotonicity of $<\alpha>$ and K respectively. Hence, $\vdash <\alpha>C \Rightarrow K<\alpha>C$.
iv) The case $B = [\alpha]C$ is proved in a similar way by using theorem 5.2.3f).
v) If $\vdash A \Rightarrow KA$ and $\vdash B \Rightarrow KB$, then $\vdash A \wedge B \Rightarrow K(A \wedge B)$ and
$\vdash A \vee B \Rightarrow K(A \vee B)$ by theorems 5.2.3b) and 5.2.3c) respectively, and tautological consequence.

DEFINITION 5.3.2

Given a formula W , the Fischer/Ladner closure of W is the smallest set of formulas FL(W) such that $W \in FL(W)$ and :

a) If $\neg A \in FL(W)$ then $A \in FL(W)$
b) If $A \vee B \in FL(W)$ then $A \in FL(W)$ and $B \in FL(W)$
c) If $[\alpha]A \in FL(W)$ then $A \in FL(W)$
d) If $[\alpha \cup \beta]A \in FL(W)$ then $[\alpha]A \in FL(W)$ and $[\beta]A \in FL(W)$
e) If $[\alpha\beta]A \in FL(W)$ then $[\alpha][\beta]A \in FL(W)$
f) If $[\alpha^*]A \in FL(W)$ then $[\alpha][\alpha^*]A \in FL(W)$
g) If $KA \in FL(W)$ then $A \in FL(W)$
h) For $\alpha = a \in A$ or $\alpha = (:e), e \in E$, if $[\alpha]A_1, [\alpha]A_2, \ldots, [\alpha]A_n$ $(n \geq 1)$ are in FL(W) s.t. A_i $(i \leq n)$ is a non K-formula and $A_i \neq A_j$ for $i \neq j$,
then $[\alpha]\neg K\neg(A_1 \wedge \ldots \wedge A_n) \in FL(W)$
i) If $(\rightarrow e) \in FL(W)$ then $K(\rightarrow e) \in FL(W)$

Given a formula W , let $FL^0(W)$ denote the smallest set of formulas containing W and closed under the rules a) to f) , $FL^1(W)$ the closure of $FL^0(W)$ under the rules a) to g) , and $FL^2(W)$ the closure of $FL^1(W)$ under the rules a) to h).

$A(W) = \{a \mid a \in A , [a]B \in FL^1(W)$ for a non K-formula $B\}$

$E(W) = \{(:e) \mid e \in E , [(:e)]B \in FL^1(W)$ for a non K-formula $B\}$

and $\forall \alpha \in A(W) \cup E(W)$, $Z(\alpha) = \{ A \mid A$ is a non K-formula and $[\alpha]A \in FL^1(W)\}$

We denote by $CZ(\alpha)$ the set of all finite conjunctions of formulas in $Z(\alpha)$.

LEMMA 5.3.2

a) Let $X_0 = FL^0(W)$ and for $i \geq 1$,

if for every formula A , $KA \notin X_{i-1}$, then $X_i = \emptyset$;

otherwise , $X_i = \bigcup(FL^0(A) \mid KA \in X_{i-1})$.

1) $\forall i \geq 0$, X_i is finite.

2) $\exists n \geq 1$ s.t. $\forall i \geq n$ $X_i = \emptyset$

3) $FL^2(W)$ is finite

b) $FL(W)$ is finite.

PROOF

a) 1) By induction on $i \geq 0$

i) For the case $i=0$, $X_0 = FL^0(W)$ and it has been proved in [FL76], [G87] and [KT89] that for any formula A , $FL^0(A)$ is finite. In fact, it is shown that the size of $FL^0(W)$ is linear in the length of W.

ii) If X_i is finite, then $\{ A \mid KA \in X_i\}$ is finite. If $\{A \mid KA \in X_i\} \neq \emptyset$, then $X_{i+1} = \bigcup(FL^0(A) \mid KA \in X_i)$ and is finite since $FL^0(A)$ is finite for any formula A. Otherwise, $X_{i+1} = \emptyset$ and is finite.

2) For any formula $A \in X_i$, let $n(A,i,K)$ denote the number of K's in A. Since X_i is finite, there is $n(i,K)$ s.t. for any formula $A \in X_i$, $n(A,i,K) \leq n(i,K)$. We also know by definition of X_i that $\forall i \geq 1, j \geq i-1$, $n(j,K) = 0$ or $n(i,K) \leq n(i-1,K)-1$. Therefore, for $i \geq n(0,K)$, $n(i,K) = 0$. Hence , $X_i = \emptyset$ for $i \geq n(0,K)+1$.

3) For any formula A , $FL^0(A)$ is closed under the rules a) to f).

Therefore $\forall i \geq 0$, X_i is closed under the rule a) to f) and their union is closed under the rules a) to g). Hence, $FL^1(W) \subseteq \bigcup_{i \geq 0} X_i$ and from the above proof it follows that $FL^1(W)$ is a subset of a finite union of finite sets. Therefore, $FL^1(W)$ is finite. Hence, $\forall \alpha \in A(W) \cup E(W)$, $Z(\alpha)$ and $CZ(\alpha)$ are finite.

Since $FL^1(W)$ is closed under the rules a) to g) and rule h) can be applied only to formulas in $CZ(\alpha)$, $\alpha \in A(W) \cup E(W)$, $FL^2(W)$ is obtained from $FL^1(W)$ by adding

to $FL^1(W)$ the formulas $[\alpha]\neg K\neg A$, $\alpha \in A(W) \cup E(W)$, and $A \in CZ(\alpha)$, and then taking their closure under rules a) to h). But $\forall \alpha \in A(W) \cup E(W)$, $FL^1(W) \cup CZ(\alpha)$ is closed under the rules a) to g). Hence, $\forall \alpha \in A(W) \cup E(W)$ and $A \in CZ(\alpha)$, at most four formulas are added to $FL^1(W) \cup CZ(\alpha)$. Therefore, $FL^2(W)$ is finite since $FL^1(W)$ and $A(W) \cup E(W)$ are finite and $\forall \alpha \in A(W) \cup E(W)$, $CZ(\alpha)$ is finite.

b) $FL(W)$ is the closure of $FL^2(W)$ under rule i) and for each formula in $FL^2(W)$, rule i) adds at most one formula in $FL^2(W)$. Therefore, $FL(W)$ is finite since $FL^2(W)$ is finite.

Note that if we denote by W' the formula obtained from W by deleting all occurences of the knowledge operator K, then the size of $FL^1(W)$ is linear in the size of $FL^0(W')$ and that of $FL(W)$ is in the worst case exponential in that of $FL^0(W')$. But we believe that it is possible to modify rule h) such that the size of $FL(W)$ is linear in that of $FL(W')$.

For a consistent formula W and PLKA-maximal sets s, t, we define the relation $s \equiv_W t$ iff for any formula $A \in FL(W)$, $A \in s$ iff $A \in t$. It is obvious that $\equiv_W$ is an equivalence relation on MS(PLKA). The equivalence class of s is denoted by $|s|$.

LEMMA 5.3.3

The set of equivalence classes $\{|s| \mid s \in MS(PLKA)\}$ is finite.

PROOF

For any PLKA-maximal set s, if $s(W) = \{ A \mid A \in s \cap FL(W)\}$, then by definition of $\equiv_W$, $|s| = |t|$ iff $s(W) = t(W)$. But since $s(W)$ is a subset of $FL(W)$ which is finite by lemma 5.3.2b), there is a maximum of 2^n equivalence classes, where n is the size of $FL(W)$.

LEMMA 5.3.4

Given a consistent formula W and a PLKA-maximal set s, let C_s denote the conjunction of all formulas in the set

$$\{ A \mid A \in s \cap FL(W) \} \cup \{ \neg A \mid A \in FL(W), \neg A \in s \}.$$

a) For any PLKA-maximal set t, $C_s \in t$ iff $|t| = |s|$ iff $C_s = C_t$.

b) For any non empty subset $X = \{|s_1|, \ldots, |s_n|\}$ of PLKA-maximal set equivalence classes, if $Cx = \bigvee (C_s \mid |s| \in X)$,
then for any PLKA-maximal set t, $|t| \in X$ iff $Cx \in t$.

c) For any formula $A \in FL(W)$, $A \in s$ iff $\vdash C_s \Rightarrow A$

d) For any formula $A \in FL(W)$, $\vdash C_s \Rightarrow A$ iff $\nvdash C_s \Rightarrow \neg A$

e) If $A, B \in FL(W)$, then $\vdash C_s \Rightarrow A \vee B$ iff $\vdash C_s \Rightarrow A$ or $\vdash C_s \Rightarrow B$

PROOF

a) It follows from the definition of $\equiv_W$ that $|s| = |t|$ iff
$\{A \mid A \in s \cap FL(W)\} \cup \{\neg A \mid A \in FL(W), \neg A \in s\} =$
$\{ A \mid A \in t \cap FL(W) \} \cup \{ \neg A \mid A \in FL(W), \neg A \in t\}$ iff $C_s = C_t$ iff $C_s \in t$ by proposition 5.3.2d).

b) $Cx \in t$ iff $C_s \in t$ for some $|s| \in X$ by proposition 5.3.2e);
iff $|t| \in X$ by a).

c) If $A \in s$ then A is a conjunct of C_s. Therefore, $\vdash C_s \Rightarrow A$ by tautological consequence. On the other hand, if $\vdash C_s \Rightarrow A$ then $A \in s$ by a) and proposition 5.3.2f).

d) $\vdash C_s \Rightarrow A$ implies $A \in s$ by c). Then, $\neg A \notin s$ by proposition 5.3.2a).
Hence, $\nvdash C_s \Rightarrow \neg A$ by proposition 5.3.2f) since $C_s \in s$ by a).
On the other hand, if $\nvdash C_s \Rightarrow \neg A$ then $\neg A$ is not a conjunct of C_s.
Therefore, $\neg A \notin s$. Hence, $\vdash C_s \Rightarrow A$ by proposition 5.3.2a) and part c).

e) $\vdash C_s \Rightarrow A \vee B$ iff $A \vee B \in s$ by c); iff $A \in s$ or $B \in s$ by proposition 5.3.2e); iff $\vdash C_s \Rightarrow A$ or $\vdash C_s \Rightarrow B$ by c).

DEFINITION OF THE MODEL

For a consistent formula W , we define the sets E(W) and Kf(W) as follows:

$$E(W) = \{ e \mid (\neg e) \in FL(W) \} \text{ and } Kf(W) = Kf \cap FL(W)$$

I.e. Kf(W) is the set of all K-formulas in FL(W). We now define the object system O(W) as follows:

$$Q = \{ |s| \mid s \in MS(PLKA) \} \text{ and } Q_{\_} = \mathcal{P}(Kf(W))$$

I.e. Q is the set of PLKA-maximal set equivalence classes, and $Q_{\_}$ is the set of all subsets of K-formulas in FL(W). The set $Q_{\_}$ is useful to define the right equivalence relation on Q.

We also want the object to be able to initiate communications to its environment only with respect to the external properties in E(W). The relations R_a , S_e and S_e- are defined as follows:

$\forall a \in A$, $a \neq \epsilon$, $(|s|, |t|) \in R_a$ iff

if $[a]A \in FL(W)$ then $\vdash C_s \Rightarrow [a]A$ implies $\vdash C_t \Rightarrow A$

$\forall e \in E$,

$(u, |s|, |t|) \in S_e$ iff $u = \emptyset$ and if $[(:e)]A \in FL(W)$

then $\vdash C_s \Rightarrow [(:e)]A$ implies $\vdash C_t \Rightarrow A$

if $e \in E(W)$, then $S_e\text{-} = \{ (|s| , \emptyset , s \cap Kf(W)) \mid \vdash C_s \Rightarrow (\neg e) \}$

otherwise, $S_e\text{-} = \emptyset$

REMARK If the set $\{ e \mid (\neg e) \in FL(W) \} = \emptyset$, then we set E(W) to $\{e_0\}$ where e_0 is any external property in E.

In this case, $S_{e_0}\text{-} = \{ (|s|, \emptyset, s \cap Kf(W)) \mid |s| \in Q \}$

Let F denote the structure $(O(W) , \sim_Q)$ and V the interpretation of the atomic formulas s.t. $\forall p \in \Phi$ and $|s| \in Q$, $|s| \in V(p)$ iff $p \in FL(W)$ and $\vdash C_s \Rightarrow p$. M is the corresponding model. We would like to prove that M is a model of W. But, we must first find out what is the behaviorally indistinguishable relation in Q. We will do that after the proof of the following theorem.

LEMMA 5.3.5

s, t are PLKA-maximal sets and $\alpha \in ACT$.

a) If $\vdash C_s \Rightarrow [\alpha^*]A$, then $\vdash C_s \Rightarrow A$ and $\vdash C_s \Rightarrow [\alpha][\alpha^*]A$.

b) If for any formula B s.t $[\alpha]B \in FL(W)$, $\vdash C_s \Rightarrow [\alpha]B$ implies $\vdash C_t \Rightarrow B$ then for any formula A s.t. $[\alpha^*]A \in FL(W)$,

$\vdash C_s \Rightarrow [\alpha^*]A$ implies $\vdash C_t \Rightarrow [\alpha^*]A$.

PROOF

a) If $\vdash C_s \Rightarrow [\alpha^*]A$ then $\vdash C_s \Rightarrow A \wedge [\alpha][\alpha^*]A$ by theorem 5.2.3q) and tautological consequence. Therefore , $\vdash C_s \Rightarrow A$ and $\vdash C_s \Rightarrow [\alpha][\alpha^*]A$ by tautological consequence.

b) If $\vdash C_s \Rightarrow [\alpha^*]A$ then $\vdash C_s \Rightarrow [\alpha][\alpha^*]A$ by part a).

But since $[\alpha^*]A \in FL(W)$, it follows that $[\alpha][\alpha^*]A \in FL(W)$ by closure rule f). Therefore , $\vdash C_t \Rightarrow [\alpha^*]A$ by hypothesis.

THEOREM 5.3.1

If $\alpha \in ACT$ and s, t are PLKA-maximal sets such that $|s|\underline{\alpha}|t|$,

then for any formula A s.t. $[\alpha]A \in FL(W)$, if $\vdash C_s \Rightarrow [\alpha]A$, then $\vdash C_t \Rightarrow A$.

PROOF By induction on α

i) The initial case $\alpha = a \in A$, $a \neq \epsilon$ or $\alpha = (:e)$, $e \in E$ follows from the definitions of R_a and S_e.

The case $\alpha = \epsilon$ follows from theorem 5.2.3a) and tautological consequence.

ii) Let $[\alpha\beta]A \in FL(W)$ s.t. $\vdash C_s \Rightarrow [\alpha\beta]A$. Then , $\vdash C_s \Rightarrow [\alpha][\beta]A$ by theorem 5.2.3o) and tautological consequence. And since $|s|\underline{\alpha\beta}|t|$, there is a PLKA-maximal set u s.t. $|s|\underline{\alpha}|u|$ and $|u|\underline{\beta}|t|$.

Therefore , $\vdash C_u \Rightarrow [\beta]A$ by closure rule e) and induction hypothesis; and by closure rule c) and induction hypothesis , $\vdash C_t \Rightarrow A$.

iii) Let $[\alpha \cup \beta]A \in FL(W)$ s.t. $\vdash C_s \Rightarrow [\alpha \cup \beta]A$. Then $\vdash C_s \Rightarrow [\alpha]A \wedge [\beta]A$ by theorem 5.2.3n) and tautological consequence.

Therefore , $\vdash C_s \Rightarrow [\alpha]A$ and $\vdash C_s \Rightarrow [\beta]A$ by tautological consequence. But since $|s| \underline{\alpha \cup \beta} |t|$, $|s| \underline{\alpha} |t|$ or $|s| \underline{\beta} |t|$. Therefore, $\vdash C_t \Rightarrow A$ by closure rule d) and induction hypothesis.

iv) Let $[\alpha^*]A \in FL(W)$ s.t. $\vdash C_s \Rightarrow [\alpha^*]A$.

$|s| \underline{\alpha^*} |t|$ implies that there exist $u_0=s$, ..., $u_n=t$ $(n \geq 0)$ s.t. $\forall i>0$, $|u_{i-1}| \underline{\alpha} |u_i|$.

If $n = 0$, then $|s| = |t|$; and $\vdash C_t \Rightarrow A$ by lemmas 5.3.4a) and 5.3.5a).

If $n>0$, then by induction hypothesis and lemma 5.3.5b), if $\vdash C_{u_{i-1}} \Rightarrow [\alpha^*]A$, then $\vdash C_{u_i} \Rightarrow [\alpha^*]A$, $\forall i>0$. But since $\vdash C_{u_0} \Rightarrow [\alpha^*]A$, it follows that $\vdash C_{u_i} \Rightarrow [\alpha^*]A$, $\forall i>0$. Therefore , $\vdash C_t \Rightarrow [\alpha^*]A$ and by lemma 5.3.5a) , $\vdash C_t \Rightarrow A$.

We will now define an equivalence relation $\sim$ on Q , and then show that it is exactely the behaviorally indistinguishable relation. In fact, if $\sim_0$, ..., $\sim_n$, ... is the sequence of equivalence relations used in the definition of $\sim_Q$, then we show that $|s| \sim |t|$ iff $|s| \sim_0 |t|$.

Let $Kf(C_s)$ denote the set:

$$\{ A \mid A \in Kf(W), \vdash C_s \Rightarrow A \} \cup \{ \neg A \mid A \in Kf(W), \vdash C_s \Rightarrow \neg A \}.$$

We define the relation $|s| \sim |t|$ on Q as follows:

$$|s| \sim |t| \text{ iff } Kf(C_s) = Kf(C_t).$$

I.e. $|s| \sim |t|$ iff for any K-formula $A \in FL(W)$, $\vdash C_s \Rightarrow A$ iff $\vdash C_t \Rightarrow A$.

Observe that $\sim$ is an equivalence relation on Q and that:

$$Kf(C_s) \subseteq t \text{ iff } Kf(C_s) = Kf(C_t) \text{ iff } s \cap Kf(W) = t \cap Kf(W).$$

LEMMA 5.3.6

a) If A is a K-formula such that $[\alpha]A \in FL(W)$ and $|s|$, $|t|$ and $|u|$ are s.t. $|s| \sim |t|$ and $|s| \underline{\alpha} |u|$, then $\vdash C_t \Rightarrow [\alpha]A$ implies $\vdash C_u \Rightarrow A$.

b) If A_1 , ..., A_n , $n \geq 1$ are non K-formulas and $\alpha = a \in A$ or $\alpha = (:e)$, $e \in E$ s.t. $\forall i \leq n$, $[\alpha]A_i \in FL(W)$ and $|s|$, $|t|$ and $|u|$ are s.t. $|s| \sim |t|$ and $|s| \underline{\alpha} |u|$, then $\vdash C_t \Rightarrow [\alpha]A_i$, $\forall i \leq n$ implies $\vdash C_u \Rightarrow \neg K \neg (A_1 \wedge \ldots \wedge A_n)$

PROOF

a) Since A is a K-formula, $[\alpha]A \in Kf(W)$ and since $\vdash C_t \Rightarrow [\alpha]A$ and $|s| \sim |t|$, it follows that $C_s \Rightarrow [\alpha]A$. Therefore , $\vdash C_u \Rightarrow A$ by theorem 5.3.1.

b) $\vdash A_1 \wedge \ldots \wedge A_n \Rightarrow \neg K \neg (A_1 \wedge \ldots \wedge A_n)$ by axiom 3 and tautological consequence. Therefore, $\vdash [\alpha]A_1 \wedge \ldots \wedge [\alpha]A_n \Rightarrow [\alpha]\neg K \neg (A_1 \wedge \ldots \wedge A_n)$ by monotonocity of $[\alpha]$, theorem 5.2.3p) and tautological consequence.

Hence $\vdash C_t \Rightarrow [\alpha]\neg K \neg (A_1 \wedge \ldots \wedge A_n)$ by tautological consequence.

Therefore , $\vdash C_u \Rightarrow \neg K \neg (A_1 \wedge \ldots \wedge A_n)$ by closure rule h) and part a).

THEOREM 5.3.2

If s , t are PLKA-maximal sets s.t. $|s| \sim |t|$, then:

a) $\forall e \in E$, $v , w \in Q_{-}$, $(|s|,v,w) \in S_e$- iff $(|t|,v,w) \in S_e$-

b) $\forall a \in A$, if $\exists |u|$ s.t. $(|s|,|u|) \in R_a$,

then $\exists |u'| \sim |u|$ s.t. $(|t|,|u'|) \in R_a$, and vice versa.

c) $\forall e \in E$ and $v \in Q_{-}$, if $\exists |u|$ s.t. $(v,|s|,|u|) \in S_e$

then $\exists |u'| \sim |u|$ s.t. $(v,|t|,|u'|) \in S_e$, and vice versa.

PROOF

a) Let $e \in E$ and $v , w \in Q_{-}$ s.t. $(|s|,v,w) \in S_e$-.

i) If $\{e \mid (\rightarrow e) \in FL(W)\} = \emptyset$, then $e = e_0$ and $(|s|,v,w) \in S_e$- iff $v = \emptyset$ and $w = s \cap Kf(W)$ by definition of S_e- ; iff $v = \emptyset$ and $w = t \cap Kf(W)$ (since $|s| \sim |t|$); iff $(|t|,v,w) \in S_e$-.

ii) If $\{e \mid (\rightarrow e) \in FL(W)\} \neq \emptyset$, then $(|s|,v,w) \in S_e$- iff $\vdash C_s \Rightarrow (\rightarrow e)$, $v = \emptyset$ and $w = s \cap Kf(W)$ by definition of S_e-; iff $\vdash C_s \Rightarrow K(\rightarrow e)$, $v = \emptyset$ and $w = s \cap Kf(W)$ by axiom 2 and tautological consequence; iff $\vdash C_t \Rightarrow K(\rightarrow e)$, $v = \emptyset$ and $w = t \cap Kf(W)$ by closure rule i) and definition of $\sim$; iff $\vdash C_t \Rightarrow (\rightarrow e)$, $v = \emptyset$ and $w = t \cap Kf(W)$ by axiom 2 and tautological consequence ; iff $(|t|,v,w) \in S_e$-.

b) Let $a \in A$ and $|u|$ s.t. $(|s|,|u|) \in R_a$ and,

Let $X = Kf(C_u) \bigcup \{ A \mid [a]A \in FL(W), \vdash C_t \Rightarrow [a]A \}$.

If X is inconsistent, then there exist $X_1, \ldots, X_n$ $(n \geq 1)$, s.t. $X_i \in Kf(C_u)$, $(\forall i \leq n)$, K-formulas $B_1, \ldots, B_l$ $(l \geq 0)$ s.t. $[a]B_i \in FL(W)$ and $\vdash C_t \Rightarrow [a]B_i$, $(\forall i \leq l)$, and non K-formulas $A_1, \ldots, A_m$ $(m \geq 1)$ s.t. $[a]A_i \in FL(W)$ and $\vdash C_t \Rightarrow [a]A_i$, $(\forall i \leq m)$ s.t. $\vdash X_1 \wedge \ldots \wedge X_n \wedge B_1 \wedge \ldots \wedge B_l \Rightarrow \neg(A_1 \wedge \ldots \wedge A_m)$.

And by monotonicity of K, lemma 5.3.1 and tautological consequence,

$\vdash X_1 \wedge \ldots \wedge X_n \wedge B_1 \wedge \ldots \wedge B_l \Rightarrow K\neg(A_1 \wedge \ldots \wedge A_m)$. But since $\vdash C_u \Rightarrow X_i$, $\forall i \leq n$ and $\vdash C_u \Rightarrow B_j$, $\forall j \leq l$ by lemma 5.3.6a), it follows by tautological consequence that $\vdash C_u \Rightarrow K\neg(A_1 \wedge \ldots \wedge A_m)$; and this is impossible by lemma 5.3.4d), since by lemma 5.3.6b), we also have $\vdash C_u \Rightarrow \neg K\neg(A_1 \wedge \ldots \wedge A_m)$. Therefore, X is consistent and by Lindenbaum's lemma there is a PLKA-maximal set u' s.t. $X \subseteq u'$.

Hence, $\exists |u'| \sim |u|$ s.t. for any formula $[a]A \in FL(W)$, $\vdash C_t \Rightarrow [a]A$ implies $\vdash C_{u'} \Rightarrow A$ by definition of $\sim$, closure rule c) and lemma 5.3.4c).

Therefore, $\exists |u'| \sim |u|$ s.t $(|t|, |u'|) \in R_a$ by definition of R_a.

The other case is proved in the same way, since the relation $\sim$ is symmetric.

c) Let $e \in E$, $v \in Q$ and $|u|$ s.t. $(v, |s|, |u|) \in S_e$.

By definition of S_e, it follows that $v = \emptyset$ and we prove the same way as in b) that $\exists |u'| \sim |u|$ s.t. for any formula $[(:e)]A \in FL(W)$, $\vdash C_t \Rightarrow [(:e)]A$ implies $\vdash C_{u'} \Rightarrow A$. Therefore, $\exists |u'| \sim |u|$ s.t. $(v, |t|, |u'|) \in S_e$ by definition of S_e.

The other case is proved in the same way, since the relation $\sim$ is symmetric.

COROLLARY 5.3.1

The states $|s|, |t| \in Q$ are behaviorally indistinguishable $(|s| \sim_Q |t|)$ iff $|s| \sim |t|$.

PROOF

Let $\sim_0, \ldots, \sim_n, \ldots$ denote the sequence of equivalence relations used in the definition of $\sim_Q$. Let's prove that $|s| \sim_0 |t|$ iff $|s| \sim |t|$.

i) If $|s| \sim |t|$, then by part a) of the theorem, $\forall e \in E$, $v, w \in Q_-$, $(|s|, v, w) \in S_{e^-}$ iff $(|t|, v, w) \in S_{e^-}$. Therefore, $|s| \sim_0 |t|$.

ii) $|s| \sim_0 |t|$ iff $\forall e \in E$, $v, w \in Q_-$, $(|s|, v, w) \in S_{e^-}$ iff $(|t|, v, w) \in S_{e^-}$, which implies that $\forall w$, $w = s \cap Kf(W)$ iff $w = t \cap Kf(W)$.

Therfore, $s \cap Kf(W) = t \cap Kf(W)$. Hence, $|s| \sim |t|$.
It then follows from parts b) and c) of the above theorem that $\forall n \geq 0$,
$|s| \sim_n |t|$ iff $|s| \sim |t|$. Hence , $|s| \sim_Q |t|$ iff $|s| \sim |t|$.

LEMMA 5.3.7

If s and $u_0 = t$ are PLKA-maximal sets and $\alpha \in ACT$ s.t.
$\{ A \mid [\alpha^*]A \in s \} \subseteq t$ then:

a) $\forall m \geq 0$, if $\forall j \leq m$, $|u_j| \neq |s|$ then $\exists u_{m+1}$ s.t. $\forall i \leq m$, $|u_{m+1}| \neq |u_i|$,
$\{ A \mid [\alpha^*]A \in s \} \subseteq u_{m+1}$ and for some $k \leq m$, $<\alpha>Cu_k \in u_{m+1}$

b) There are PLKA-maximal sets $v_0 = s , v_1 , \ldots, v_n = t$ $(n \geq 0)$ s.t.
$\forall i > 0$, $<\alpha>Cv_i \in v_{i-1}$.

PROOF

a) By induction on $m \geq 0$

i) For the case $m = 0$, $\{A \mid [\alpha^*]A \in s\} \subseteq t$ and let's assume that $|t| \neq |s|$.
Then by lemma 5.3.4a) and proposition 5.3.3a), $<\alpha^*>Ct \in s$.
Therefore, $Ct \vee <\alpha^*>(\neg Ct \wedge <\alpha>Ct) \in s$ by axiom 12 and proposition 5.3.2f).
Hence, $Ct \in s$ or $<\alpha^*>(\neg Ct \wedge <\alpha>Ct) \in s$ by proposition 5.3.2e). But since $|t| \neq |s|$, $Ct \notin s$ by lemma 5.3.4a). Therefore, $<\alpha^*>(\neg Ct \wedge <\alpha>Ct) \in s$; and by proposition 5.3.3a) there is a PLKA-maximal set u_1 s.t. $\{A \mid [\alpha^*]A \in s\} \subseteq u_1$ and $\neg Ct \wedge <\alpha>Ct \in u_1$. Hence, there is a PLKA-maximal set u_1 s.t.
$\{A \mid [\alpha^*]A \in s\} \subseteq u_1$, $|u_1| \neq |t|$ and $<\alpha>Ct \in u_1$ by propositions 5.3.2d) and 5.3.2a) and lemma 5.3.4a).

ii) Let's assume the hypothesis true for m-1 and that $\forall j \leq m-1$, $|u_j| \neq |s|$.
Then by induction hypothesis, $\exists u_m$ s.t. $\forall i \leq m-1$, $|u_m| \neq |u_i|$, $\{A \mid [\alpha^*]A \in s\} \subseteq u_m$ and for some $k \leq m-1$, $<\alpha>Cu_k \in u_m$. Hence, by lemma 5.3.4a) and proposition 5.3.3a) $\exists u_m$ s.t. $\forall i \leq m-1$ $|u_m| \neq |u_i|$, $<\alpha^*>Cu_m \in s$ and for some $k \leq m-1$, $<\alpha>Cu_k \in u_m$. From proposition 5.3.2e), axiom 10 and proposition 5.3.2f), it follows that $<\alpha^*>(Cu_m \vee \ldots \vee Cu_0) \in s$.

Therefore,

$Cu_m \vee \ldots \vee Cu_0 \in s$ or $<\alpha^*>(\neg(Cu_m \vee \ldots \vee Cu_0) \wedge <\alpha>(Cu_m \vee \ldots \vee Cu_0)) \in s$ by axiom 12 , propositions 5.3.2f) and e).

If $|u_m| \neq |s|$, then by lemma 5.3.4b) $Cu_m \vee \ldots \vee Cu_0 \notin s$ since $\forall j \leq m\ |u_j| \neq |s|$. Therefore, $<\alpha^*>(\neg(Cu_m \vee \ldots \vee Cu_0) \wedge <\alpha>(Cu_m \vee \ldots \vee Cu_0)) \in s$; and $\exists u_{m+1}$ s.t. $\{A \mid [\alpha^*]A \in s\} \subseteq u_{m+1}$, $\neg(Cu_m \vee \ldots \vee Cu_0) \in u_{m+1}$ and $<\alpha>(Cu_m \vee \ldots \vee Cu_0) \in u_{m+1}$ by propositions 5.3.3a) and 5.3.2d). Therefore , $\exists u_{m+1}$ s.t. $\{A \mid [\alpha^*]A \in s\} \subseteq u_{m+1}$, $|u_{m+1}| \neq |u_i|$, $\forall i \leq m$ and $<\alpha>Cu_m \vee \ldots \vee <\alpha>Cu_0 \in u_{m+1}$ by Proposition 5.3.2a), lemma 5.3.4b), axiom 10, and proposition 5.3.2f). And by proposition 5.3.2e), $<\alpha>Cu_m \vee \ldots \vee <\alpha>Cu_0 \in u_{m+1}$ implies that for some $k \leq m$, $<\alpha>Cu_k \in u_{m+1}$.

b) It follows from a) that for some $m \geq 0$, $|u_m| = |s|$ and that there is a sequence of distinct PLKA-maximal set equivalence classes $|u_0| = |t|$, $|u_1|, \ldots, |u_m| = |s|$ s.t. $\forall j > 0$, $\exists k < j$ s.t. $<\alpha>Cu_k \in u_j$.

If there is no such m , then the sequence is infinite; which is impossible, since the set of PLKA-maximal set equivalence classes is finite. Let $v_0 = u_m = s$, and for $i > 0$, $v_i = u_k$ s.t. $<\alpha>Cu_k \in v_{i-1}$. It follows from a) that for some $n \geq 0$, $v_n = t$.

THEOREM 5.3.3

If s , t are PLKA-maximal sets and $\alpha \in ACT$ s.t. $\{ A \mid [\alpha]A \in s \} \subseteq t$, then $|s| \underline{\alpha} |t|$.

PROOF By induction on α

i) For the initial case $\alpha = a \in A$, $a \neq \epsilon$ or $\alpha = (:e)$, $e \in E$, let $[\alpha]A \in FL(W)$ s.t. $\vdash C_s \Rightarrow [\alpha]A$. Then $[\alpha]A \in s$ by lemma 5.3.4c) and by hypothesis, $A \in t$. Therefore, $\vdash C_t \Rightarrow A$ by closure rule c) and lemma 5.3.4c). Hence, $|s| \underline{\alpha} |t|$.

For the case $\alpha = \epsilon$, if $\{A \mid [\epsilon]A \in s\} \subseteq t$, then $s \subseteq t$ by theorem 5.2.3a) and proposition 5.3.2f). Hence, $|s| = |t|$ by lemma 5.3.4a). Therefore, $|s| \underline{\alpha} |t|$.

ii) The cases $\alpha = \beta\gamma$ and $\alpha = \beta \cup \gamma$ follows from propositions 5.3.4a) and 5.3.4b) respectively.

iii) If $\{ A \mid [\beta^*]A \in s\} \subseteq t$, then by lemma 5.3.7b), there are PLKA-maximal sets $v_0=s, v_1, \ldots, v_n=t$ $(n \geq 0)$ s.t. $\forall i>0$, $<\beta>Cv_i \in v_{i-1}$. It then follows from proposition 5.3.3a) that there are PLKA-maximal sets $v_0=s, v_1, \ldots, v_n=t$ $(n \geq 0)$ s.t. $\forall i>0$, $\exists v'_i$ s.t. $\{A \mid [\beta]A \in v_{i-1}\} \cup \{Cv_i\} \subseteq v'_i$. But, $Cv_i \in v'_i$ implies that $|v_i| = |v'_i|$ by lemma 5.3.4a).

Therefore, there are PLKA-maximal sets $v_0=s, v_1, \ldots, v_n=t$ $(n \geq 0)$ s.t. $\forall i>0$, $|v_{i-1}| \underline{\beta} |v_i|$ by induction hypothesis. Hence, $|s| \underline{\beta^*} |t|$.

LEMMA 5.3.8

For any formula $A \in FL(W)$ and $|s| \in Q$, $M,|s| \models A$ iff $\vdash C_s \Rightarrow A$.

PROOF By induction on the structure of A.

a) The initial case $p \in \Phi$ follows from the definition of M; and for the case $A = (\rightarrow e), e \in E$, $M,|s| \models (\rightarrow e)$ iff $(|s|, \emptyset, s \cap Kf(W)) \in S_e$- iff $\vdash C_s \Rightarrow (\rightarrow e)$ by definition of S_e-.

b) If $A = \neg B$, then $M,|s| \models \neg B$ iff $M,|s| \not\models B$ iff $\not\vdash C_s \Rightarrow B$ by closure rule a) and induction hypothesis; iff $\vdash C_s \Rightarrow \neg B$ by lemma 5.3.4d).

c) If $A = B \vee C$, then $M,|s| \models B \vee C$ iff $M,|s| \models B$ or $M,|s| \models C$ iff $\vdash C_s \Rightarrow B$ or $\vdash C_s \Rightarrow C$ by closure rule b) and induction hypothesis;
iff $\vdash C_s \Rightarrow A \vee B$ by lemma 5.3.4e).

d) Let $A = KB$.

i) It follows from the definition of $\sim$ that if $\vdash C_s \Rightarrow KB$, then $\forall |t| \sim |s|$, $\vdash C_t \Rightarrow KB$. Therefore, $\forall |t| \sim |s|$, $\vdash C_t \Rightarrow B$ by axiom 3 and tautological consequence. Hence, $\forall |t| \sim |s|$, $M,|t| \models B$ by closure rule g) and induction hypothesis. Therefore, $M,|s| \models KB$.

ii) Let's assume that $\not\vdash C_s \Rightarrow KB$ and let $X = Kf(C_s) \cup \{\neg B\}$.
If X is inconsistent, then there exist $X_1, \ldots, X_n$, $n \geq 1$, $X_i \in Kf(C_s)$, $\forall i \leq n$ s.t. $\vdash X_1 \wedge \ldots \wedge X_n \Rightarrow B$. Therefore, $\vdash K(X_1 \wedge \ldots \wedge X_n) \Rightarrow KB$ by monotonicity of K. Hence, $\vdash X_1 \wedge \ldots \wedge X_n \Rightarrow KB$ by lemma 5.3.1 and tautological consequence.

And by tautological consequence, $\vdash C_u \Rightarrow KB$. (Contradiction). Therefore, X is consistent and by Lindenbaum's lemma, there is a PLKA-maximal set t s.t. $X \subseteq t$. Hence, $\exists |t| \sim |s|$ s.t. $\nvdash C_t \Rightarrow B$ by proposition 5.3.2a) and lemma 5.3.4c). Therefore, $\exists |t| \sim |s|$ s.t. $M, |t| \nvdash B$ by closure rule g) and induction hypothesis. So, $M, |s| \nvdash KB$.

e) Let $A = [\alpha]B$.

i) Let's assume that $\vdash C_s \Rightarrow [\alpha]B$ and let $|t|$ s.t. $|s| \underline{\alpha} |t|$.

Then $\vdash C_t \Rightarrow B$ by theorem 5.3.1, and $M, |t| \models B$ by closure rule c) and induction hypothesis. Hence, $M, |s| \models [\alpha]B$.

ii) If $\nvdash C_s \Rightarrow [\alpha]B$ then $<\alpha>\neg B \in s$ by lemma 5.3.4c) and proposition 5.3.2a). Therefore, there is a PLKA-maximal set t s.t. $\{A \mid [\alpha]A \in s\} \cup \{\neg B\} \subseteq t$ by proposition 5.3.3a). And by theorem 5.3.3, proposition 5.3.2a) and lemma 5.3.4c), there is a PLKA-maximal set t s.t. $|s| \underline{\alpha} |t|$ and $\nvdash C_t \Rightarrow B$. Therefore, there is a PLKA-maximal set t s.t. $|s| \underline{\alpha} |t|$ and $M, |t| \nvdash B$ by closure rule c) and induction hypothesis. Hence , $M, |s| \nvdash [\alpha]B$.

THEOREM 5.3.4: Completeness and Decidability

PLKA is complete and decidable.

PROOF

Given a consistent formula W , $W \in FL(W)$ and by Lindenbaum's lemma, there is a PLKA-maximal set s s.t. $W \in s$. But, $W \in s$ iff $\vdash C_s \Rightarrow W$ (by lemma 5.3.4c) iff $M, |s| \models W$ by the above lemma. It also follows from lemma 5.3.3 that the model M is finite. Therefore, the set $X = \{ A \mid \models A \} = \{ A \mid \vdash A \}$ is r.e. and the set $X^c = \{ A \mid \nvdash A \} = \{ A \mid \neg A$ has a model $\} =$
$\{ A \mid \neg A$ has a finite model $\}$ is r.e. Since X, X^c are both r.e., both are recursive. Hence, PLKA is complete and decidable.

5.4 PROPERTIES OF $\mathcal{L}$-FRAMES

In this section, we prove some properties of $\mathcal{L}$-frames related to behaviorally similar and epimorphic objects.

PROPOSITION 5.4.1

Suppose that O^i , O^j are object systems and f: Q^i -----> Q^j is a function s.t.

a) $\forall a \in A$ and s, t $\in Q^i$, if $(s,t) \in R^i_a$ then $(f(s),f(t)) \in R^j_a$

b) $\forall e \in E$ and s, t $\in Q^i$,

if $\exists w$ s.t. $(w,s,t) \in S^i_e$,

then $\exists w'$ s.t. $(w',f(s),f(t)) \in S^j_e$.

Then $\forall \alpha \in$ ACT and s,t $\in Q^i$, if $s \underline{\alpha} t$ then $f(s) \underline{\alpha} f(t)$.

PROOF By induction on the structure of α.

The initial case $\alpha = a \in A$ or $\alpha = (:e)$, $e \in E$ follows from the definition of f.

If we assume the hypothesis true for α and β, then:

i) $s \underline{\alpha\beta} t$ iff $\exists u$ s.t. $s \underline{\alpha} u$ and $u \underline{\beta} t$ implies $f(s) \underline{\alpha} f(u)$ and $f(u) \underline{\beta} f(t)$.

Therefore, $f(s) \underline{\alpha\beta} f(t)$.

ii) $s \underline{\alpha \cup \beta} t$ iff $s \underline{\alpha} t$ or $s \underline{\beta} t$ implies $f(s) \underline{\alpha} f(t)$ or $f(s) \underline{\beta} f(t)$.

Hence, $f(s) \underline{\alpha \cup \beta} f(t)$.

iii) $s \underline{\alpha^*} t$ iff $\exists s_0 = s, s_1, \ldots, s_n = t$, $n \geq 0$ s.t. $s_0 \underline{\alpha} s_1, \ldots, s_{n-1} \underline{\alpha} s_n$ which implies that $\exists s_0 = s, s_1, s_2, \ldots, s_n = t$, $n \geq 0$ s.t.

$f(s) \underline{\alpha} f(s_1), f(s_1) \underline{\alpha} f(s_2), \ldots, f(s_{n-1}) \underline{\alpha} f(s_n)$. Therefore, $f(s) \underline{\alpha^*} f(t)$.

PROPOSITION 5.4.2

O^i , O^j are behaviorally similar objects systems and $f^i : Q^i$ ----> Q^j , $f^j : Q^j$ ----> Q^i form a pair of behavioral similarity functions, and $\alpha \in$ ACT.

k denotes i (respectively j) iff l denotes j (respectively i).

a) If $s \underline{\alpha} t$ then $f^k(s) \underline{\alpha} f^k(t)$

b) If $f^k(s) \underline{\alpha} u$ then there is a $t \in Q^k$ s.t. $f^k(t) \sim_{Q^l} u$ and $s \underline{\alpha} t$

c) If $f^k(s) \underline{\alpha} u$ then there is a $t \in Q^k$ s.t. $f^k(t) \sim_{Q^l} u$ and $f^k(s) \underline{\alpha} f^k(t)$

PROOF:

a) Follows from proposition 5.4.1

b) If $f^k(s) \underline{\alpha} u$ then $f^l(f^k(s)) \underline{\alpha} f^l(u)$ by part a). But since $f^l(f^k(s)) \in \bar{s}$, by proposition 5.1.1 $\exists t \sim_{Q^k} f^l(u)$ s.t. $s \underline{\alpha} t$.

Therefore, $\exists t$ s.t. $f^k(t) \sim_{Q^l} u$ and $s \underline{\alpha} t$ since $t \sim_{Q^k} f^l(u)$ iff $f^k(t) \sim_{Q^l} u$.

c) follows from parts b) and a).

PROPOSITION 5.4.3

If O^i, O^j are objects and $f: Q^i \dashrightarrow Q^j$ is an object epimorphism, then:

a) $\forall s, t \in Q^i$, if $s \underline{\alpha} t$ then $f(s) \underline{\alpha} f(t)$.

b) $\forall s \in Q^i, u \in Q^j$, if $f(s) \underline{\alpha} u$ then $\exists t \in Q^i$ s.t. $f(t)=u$ and $s \underline{\alpha} t$

PROOF

a) Follows from proposition 5.4.1

b) By induction on the structure of α.

The initial case $\alpha=a \in A$ or $\alpha=(:e), e \in E$ follows from the definition of f.

If we assume the hypothesis true for α and β, then:

i) $f(s) \underline{\alpha\beta} u$ iff $\exists v$ s.t. $f(s) \underline{\alpha} v$ and $v \underline{\beta} u$ implies $\exists t_1 \in Q^i$ s.t. $f(t_1)=v$, $s \underline{\alpha} t_1$ and $f(t_1) \underline{\beta} u$. Hence, $\exists t_1, t_2 \in Q^i$ s.t. $f(t_2)=u$, $s \underline{\alpha} t_1$ and $t_1 \underline{\beta} t_2$. Therefore, $\exists t=t_2$ s.t. $f(t)=u$ and $s \underline{\alpha\beta} t$.

ii) $f(s) \underline{\alpha \cup \beta} u$ iff $f(s) \underline{\alpha} u$ or $f(s) \underline{\beta} u$ implies $\exists t_1$ s.t. $f(t_1)=u$ and $s \underline{\alpha} t_1$ or $\exists t_2$ s.t. $f(t_2)=u$ and $s \underline{\beta} t_2$. But $s \underline{\alpha} t_1$ implies $s \underline{\alpha \cup \beta} t_1$, and $s \underline{\beta} t_2$ implies $s \underline{\alpha \cup \beta} t_2$. Therefore, $\exists t$ $(t=t_1$ or $t=t_2)$ s.t. $f(t)=u$ and $s \underline{\alpha \cup \beta} t$.

iii) $f(s) \underline{\alpha^*} u$ iff $\exists s_0=f(s), s_1, \ldots, s_n=u, n \geq 0$ s.t. $f(s) \underline{\alpha} s_1, \ldots, s_{n-1} \underline{\alpha} s_n$ implies $\exists s_0=f(s), s_1, s_2, \ldots, s_n=u, n \geq 0$ and t_1 s.t. $f(t_1)=s_1$, $s \underline{\alpha} t_1$, $f(t_1) \underline{\alpha} s_2$, $s_2 \underline{\alpha} s_3, \ldots, s_{n-1} \underline{\alpha} s_n$. By successive application of the induction hypothesis on $f(t_i) \underline{\alpha} s_{i+1}$, $i \geq 1$, we obtain $t_1, t_2, \ldots, t_n$ s.t. $f(t_n)=s_n=u$, and $s \underline{\alpha} t_1$, $t_1 \underline{\alpha} t_2, \ldots, t_{n-1} \underline{\alpha} t_n$. Hence, $\exists t = t_n$ s.t. $s \underline{\alpha^*} t$ and $f(t) = u$.

In the following, F^i and F^j are frames corresponding to behaviorally similar object systems O^i and O^j respectively, and $f^i : Q^j \dashrightarrow Q^j$, $f^j : Q^j \dashrightarrow Q^j$ form a pair of behavioral similarity functions. M^i and M^j are models on F^i and F^j respectively such that for any atomic formula $p \in \Phi$, $\forall s \in Q^i$, $s \in V^i(p)$ iff $f^i(s) \in V^j(p)$ and $\forall u \in Q^j$, $u \in V^j(p)$ iff $f^j(u) \in V^i(p)$

THEOREM 5.4.1

For any formula A not containing the operator $<\alpha>$, $\alpha \in ACT$,

a) $M^i,s \models A$ iff $M^j,f^i(s) \models A$ and $M^j,u \models A$ iff $M^i,f^j(u) \models A$

b) $\forall \alpha \in ACT$, $M^i,s \models <\alpha>KA$ iff $M^j,f^i(s) \models <\alpha>KA$ and

$M^j,u \models <\alpha>KA$ iff $M^i,f^j(u) \models <\alpha>KA$.

PROOF:

a) By induction on the structure of A:

The case $A = \top$ is trivial and the cases $A = p \in \Phi$ and $A = (\rightarrow e)$, $e \in E$ follow respectively from the definitions of M^i and M^j, and the behavioral similarity relation

i) $M^i,s \models \neg A$ iff not $M^i,s \models A$ iff not $M^j,f^i(s) \models A$ (by induction hypothesis) iff $M^j,f^i(s) \models \neg A$. The case $M^j,u \models \neg A$ iff $M^i,f^j(u) \models \neg A$ is derived similarly.

ii) $M^i,s \models A \wedge B$ iff $M^i,s \models A$ and $M^i,s \models B$ iff $M^j,f^i(s) \models A$ and $M^j,f^i(s) \models B$ (by induction hypothesis) iff $M^j,f^i(s) \models A \wedge B$.

The case $M^j,u \models A \wedge B$ iff $M^i,f^j(u) \models A \wedge B$ is proved in a similar way.

iii) 1) Let's assume that $M^i,s \models KA$ and let u s.t. $u \sim_{Q^i} f^i(s)$. Then $f^j(u) \sim_{Q^i} s$. Therefore, $M^i,f^j(u) \models A$ and $M^j,u \models A$ (by induction hypothesis). Hence, $M^j,f^i(s) \models KA$.

2) Let's assume that $M^j,f^i(s) \models KA$ and let t s.t. $t \sim_{Q^j} s$. Then $f^i(t) \sim_{Q^j} f^i(s)$. Therefore, $M^j,f^i(t) \models A$ and $M^i,t \models A$ (by induction hypothesis). Hence, $M^i,s \models KA$.

$M^j,u \models KA$ iff $M^i,f^j(u) \models KA$ is proved in a similar way.

b) i) $M^i,s \models \langle\alpha\rangle KA$ iff $\exists t$ s.t. $s \underline{\alpha} t$ and $M^i,t \models KA$. Then $\exists t$ s.t. $f^i(s) \underline{\alpha} f^i(t)$ and $M^j,f^i(t) \models KA$ by proposition 5.4.2a) and part a). Therefore, $M^j,f^i(s) \models \langle\alpha\rangle KA$.

ii) $M^j,f^i(s) \models \langle\alpha\rangle KA$ iff $\exists u$ s.t. $f^i(s) \underline{\alpha} u$ and $M^j,u \models KA$.

Then $\exists u$ and t s.t. $s \underline{\alpha} t$, $f^i(t) \sim_{Qj} u$ and $M^j,u \models KA$ (by proposition 5.4.2.b). Therefore, $\exists t$ s.t. $s \underline{\alpha} t$ and $M^j,f^i(t) \models KA$. Hence, $\exists t$ s.t. $s \underline{\alpha} t$ and $M^i,t \models KA$ by part a). So, $M^i,s \models \langle\alpha\rangle KA$.

$M^j,u \models \langle\alpha\rangle KA$ iff $M^i,f^i(u) \models \langle\alpha\rangle KA$ is proved in a similar way.

COROLLARY 5.4.1

a) If f^i (respectively f^j) is an object epimorphism, then for any formula A, $M^i,s \models A$ iff $M^j,f^i(s) \models A$ (respectively, $M^j,u \models A$ iff $M^i,f^j(u) \models A$)

b) If O^i (respectively O^j) is an epimorphic image of O^j (respectively O^i), then for any formula A, $M^i \models A$ iff $M^j \models A$

PROOF:

a) It follows from the above theorem that it is enough to prove that if $M^i,s \models A$ iff $M^j,f^i(s) \models A$, then $\forall\alpha \in ACT$, $M^i,s \models \langle\alpha\rangle A$ iff $M^j,f^i(s) \models \langle\alpha\rangle A$ if f^i is an object epimorphism.

Let's assume that $M^i,s \models A$ iff $M^j,f^i(s) \models A$ and that f^i is an object epimorphism.

i) If $M^i,s \models \langle\alpha\rangle A$ then $\exists t$ s.t. $s \underline{\alpha} t$ and $M^i,t \models A$. Then by proposition 5.4.2.a), $f^i(s) \underline{\alpha} f^i(t)$ and $M^i,t \models A$. Therefore, $\exists t$ s.t. $f^i(s) \underline{\alpha} f^i(t)$ and $M^j,f^i(t) \models A$ by hypothesis. Hence, $M^j,f^i(s) \models \langle\alpha\rangle A$.

ii) If $M^j,f^i(s) \models \langle\alpha\rangle A$ then $\exists u$ s.t. $f^i(s) \underline{\alpha} u$ and $M^j,u \models A$.

Then by proposition 5.4.3b) $\exists t$ s.t. $f^i(t)=u$, $s \underline{\alpha} t$ and $M^j,f^i(t) \models A$.

Therefore, $\exists t$ s.t. $s \underline{\alpha} t$ and $M^i,t \models A$ by hypothesis. Hence, $M^i,s \models \langle\alpha\rangle A$.

b) Let's assume that O^j is an epimorphic image of O^i.

i) Let A s.t. $M^i \vdash A$ and $u \in Q^j$. Since f^i is surjective, $\exists t \in Q^i$ s.t. $f^i(t)=u$. Hence, $\exists t \in Q^i$ s.t. $f^i(t)=u$ and $M^j,t \vdash A$. Hence, $M^j,u \vdash A$ by part a). Therefore, $M^j \vdash A$.

ii) Let A s.t. $M^j \vdash A$ and $s \in Q^i$. $M^j,f^i(s) \vdash A$ since $f^i(s) \in Q^j$. Hence $M^i,s \vdash A$ by part a). Therefore, $M^i \vdash A$.

LEMMA 5.4.1: P-Morphism Lemma

If M^i and M^j are models on frames F^i and F^j respectively and the function f^i : Q^i ---> Q^j is a surjection such that $\forall s,t \in Q^i$:

a) $s \sim_{Q^i} t$ iff $f^i(s) \sim_{Q^j} f^i(t)$

b) $\forall \alpha \in ACT$, if $s \underline{\alpha} t$ then $f^i(s) \underline{\alpha} f^i(t)$

c) $\forall \alpha \in ACT$, if $f^i(s) \underline{\alpha} u$ then $\exists t \in Q^i$ s.t. $f^i(t) = u$ and $s \underline{\alpha} t$

d) $\forall p \in \Phi$, $s \in V^i(p)$ iff $f^i(s) \in V^j(p)$

Then for any formula A, $M^i,s \vdash A$ iff $M^j,f^i(s) \vdash A$.

PROOF: By induction on the structure of A.

The initial case $A=p \in \Phi$ follows from the definition of M^i and M^j and the case $(\rightarrow e)$, $e \in E$ follows from condition a).

i) $M^i,s \vdash \neg A$ iff $M^i,s \nvdash A$ iff $M^j,f^i(s) \nvdash A$ (by induction hypothesis) iff $M^j,f^i(s) \vdash \neg A$.

ii) $M^i,s \vdash A \wedge B$ iff $M^i,s \vdash A$ and $M^i,s \vdash B$ iff $M^j,f^i(s) \vdash A$ and $M^j,f^i(s) \vdash B$ (by induction hypothesis) iff $M^j,f^i(s) \vdash A \wedge B$.

iii) 1) Let's assume that $M^i,s \vdash KA$ and let u s.t. $u \sim_{Q^j} f^i(s)$. $\exists t \in Q^i$ s.t. $f^i(t)=u$ since f^i is surjective. But $f^i(t) \sim_{Q^j} f^i(s)$ iff $t \sim_{Q^i} s$ by condition a). Then, $M^i,t \vdash A$. Therefore $M^j,u \vdash A$ (by induction hypothesis). Hence, $M^j,f^i(s) \vdash KA$.

2) Let's assume that $M^j,f^i(s) \vdash KA$ and let t s.t. $t \sim_{Q^i} s$. Then $f^i(t) \sim_{Q^j} f^i(s)$ by condition a). Therefore $M^j,f^i(t) \vdash A$. Hence, $M^i,t \vdash A$ (by induction hypothesis). So, $M^i,s \vdash KA$.

iv) 1) $M^i,s \vdash <\alpha>A$ iff $\exists t$ s.t. $s \underline{\alpha} t$ and $M^i,t \vdash A$. Then $\exists t$ s.t. $f^i(s) \underline{\alpha} f^i(t)$ and $M^j,f^i(t) \vdash A$ by condition b) and induction hypothesis.

Therefore , $M^i,f^i(s) \vdash <\alpha>A$.

2) $M^j,f^i(s) \vdash <\alpha>A$ iff $\exists u$ s.t. $f^i(s) \underline{\alpha} u$ and $M^j,u \vdash A$. Then $\exists t \in Q^i$ s.t. $f^i(t)=u$, $s \underline{\alpha} t$ and $M^j,f^i(t) \vdash A$ by condition c). Therefore, $\exists t \in Q^i$ s.t. $s \underline{\alpha} t$ and $M^i,t \vdash A$ by induction hypothesis. Hence, $M^i,s \vdash <\alpha>A$.

THEOREM 5.4.2

F^i and F^j are frames based on object systems O^i and O^j respectively.

If the function f is an epimorphism from O^i to O^j that is surjective, then for any model M^i on frame F^i, there is a model M^j_i on frame F^j s.t. for any formula A, $M^i,f(s) \vdash A$ iff $M^j_i,s \vdash A$.

PROOF

For any model M^i on F^i, let M^j_i be the model on F^j s.t.

$\forall p \in \Phi$, $s \in Q^j$, $s \in V^j_i(p)$ iff $f(s) \in V^i(p)$. It follows from proposition 5.4.3 and the P-morphism lemma that for any formula A , $M^i,f(s) \vdash A$ iff $M^j_i,s \vdash A$.

COROLLARY 5.4.2

If Object system O^i is an epimorphic image of object system O^j , then for any formula A, $F^j \vdash A$ implies $F^i \vdash A$.

PROOF:

Since O^i is an epimorphic image of O^j, there is an epimorphism $f: Q^j \dashrightarrow Q^i$ that is surjective.

If not $F^i \vdash A$, then there is a model M^i on F^i and $u \in Q^i$ s.t. not $M^i, u \vdash A$. And since $\exists s \in Q^j$ s.t. $f(s)=u$, it follows from the above theorem that there is a model M^j_i on F^j s.t. not $M^j_i, s \vdash A$. Therefore, not $F^j \vdash A$.

Hence, if $F^j \vdash A$ then $F^i \vdash A$.

6. OBJECTS' SEMANTICS

In this section, we characterize objects by their knowledge-based ability. That means what they can (or cannot) achieve at any of their local states, based on what they know at that state. Notice that this notion corresponds to that of an entity's procedural knowledge introduced earlier, or that of "laws of ability" introduced in [MH69]. However, it is different from Halpern and Fagin [HF89] notion of knowledge-based protocol in the sense that an entity knowledge-based protocol specifies what it can do based on what it "knows", but says nothing about what it will achieve by doing what it actually chooses to do.

It will then be possible to provide a knowledge-based characterization of behaviorally similar objects, inheritance, and rational behavior. To this end, we first introduce the notion of an object's interpretation, and that of knowledge transformation rules.

6.1 DEFINITIONS

Given an object system O, we define an interpretation of O to be a pair (Φ, V) where Φ is a set of atomic formulas, and V a function from Φ to $\mathcal{P}(Q)$.

I.e. V is an interpretation of the atomic formulas, given the frame F.

The corresponding model is denoted by M.

Given an object system and its interpretation, a knowledge transformation rule is a formula of the form $KA \Rightarrow <\alpha> KB$ or $KA \Rightarrow [\alpha]\perp$ or $KA \Rightarrow (\rightarrow e)$ or $KA \Rightarrow \neg(\rightarrow e)$ where $\alpha \in ACT$, $\alpha \neq \epsilon$, $e \in E$ and $A \neq \perp$.

The first formula says that there is an execution of α that brings about knowledge of B whenever the object knows A, and the second says that the object is blocked with respect to α whenever it knows A, whereas the third says that the object may initiate a communication (or an external action) with respect to e whenever it knows A, and the last one says that it is impossible for the object to initiate a communication (or an external action) with respect to e whenever it knows A.

Since the formula $K(\rightarrow e) \Leftrightarrow (\rightarrow e)$ is valid for every $e \in E$, the third and the last forms can be replaced respectively by the forms $KA \Rightarrow <\epsilon>K(\rightarrow e)$ and $KA \Rightarrow <\epsilon>K\neg(\rightarrow e)$. So, a knowledge transformation rule is a formula of the form:

$$KA \Rightarrow <\alpha>KB \quad \text{or} \quad KA \Rightarrow [\alpha]\perp \quad \text{where } \alpha \in ACT,\ A \neq \perp \quad \text{and}$$

$\alpha = \epsilon$ only if $B = (\rightarrow e)$ or $B = \neg(\rightarrow e)$ for some $e \in E$.

For each local state $s \in Q$, we identify s with the conjunction of all literals (atomic formulas or negation of atomic formulas) true at s, and $\bar{s}$ with the disjunction of all such conjunctions in the equivalence class of s. I.e. $\bar{s} = \bigvee (t \mid t \sim_Q s)$.

For $c = a \in A - \{\epsilon\}$ s.t. $R_a \neq \emptyset$ or $c = (:e)$, $e \in E$ s.t. $S_e \neq \emptyset$, we define the formulas $\Gamma(\bar{s},c)$ and $\bar{\Gamma}(\bar{s},c)$ as follows:

If there is a t s.t. $s \underline{c} t$, then $\Gamma(\bar{s},c) := \bigwedge (<c>K\bar{t} \mid s \underline{c} t)$ and $\bar{\Gamma}(\bar{s},c) := \top$ otherwise, $\Gamma(\bar{s},c) := \top$ and $\bar{\Gamma}(\bar{s},c) := [c]\perp$.

Notice that $\Gamma(\bar{s},c)$ when different from $\top$ indicates all possible knowledge of facts that c can bring about from $\bar{s}$, and $\bar{\Gamma}(\bar{s},c)$ when different from $\top$ indicates that the execution of c from any state in $\bar{s}$ is impossible.

$\Gamma(\bar{s},\epsilon)$ and $\bar{\Gamma}(\bar{s},\epsilon)$ are defined as follows:

If there is no $e \in E$ s.t. $M,s \models (\rightarrow e)$ then $\Gamma(\bar{s},\epsilon) := \top$ and

$\bar{\Gamma}(\bar{s},\epsilon) := \bigwedge (<\epsilon>K\neg(\rightarrow e) \mid e \in E,\ S_e\text{-} \neq \emptyset)$;

If for every $e \in E$ s.t. $S_e\text{-} \neq \emptyset$, $M,s \models (\rightarrow e)$,

then $\Gamma(\bar{s},\epsilon) := \bigwedge (<\epsilon>K(\rightarrow e) \mid e \in E,\ S_e\text{-} \neq \emptyset)$ and $\bar{\Gamma}(\bar{s},\epsilon) := \top$;

otherwise, $\Gamma(\bar{s},\epsilon) := \bigwedge (<\epsilon>K(\rightarrow e) \mid e \in E,\ M,s \models (\rightarrow e))$ and

$$\bar{\Gamma}(\bar{s},\epsilon) := \bigwedge (<\epsilon>K\neg(\rightarrow e) \mid e \in E,\ S_e\text{-} \neq \emptyset,\ M,s \not\models (\rightarrow e)).$$

$\Gamma(\bar{s},\epsilon)$ (respectively $\bar{\Gamma}(\bar{s},\epsilon)$) when different from $\top$ indicates the types of communications (or external actions) that can be initiated from local states in $\bar{s}$ (respectively that cannot be initiated from local states in $\bar{s}$).

Let $\Gamma(\bar{s}) := \bigwedge (\Gamma(\bar{s},c) \wedge \bar{\Gamma}(\bar{s},c) \mid c = a \in A,\ R_a \neq \emptyset \text{ or } c = (:e),\ e \in E,\ S_e \neq \emptyset)$.

We denote by $\Gamma(O)$ the set of formulas $\{ K\bar{s} \Rightarrow \Gamma(\bar{s}) \mid \bar{s} \in \bar{Q} \}$

DEFINITION 6.1.1 : Procedural Knowledge

Given an interpretation of object system O , the procedural knowledge of O denoted by PK(O) is the theory of knowledge transformation rules logically implied by $\Gamma(O)$. I.e. $PK(O) = \{ A \mid \vdash \Gamma(O) \Rightarrow A$ and A is a knowledge transformation rule $\}$.

In other words, the procedural knowledge of an object tells us whether or not the execution of an activity is possible whenever the object knows a fact A; and if it is possible, what are the facts whose knowledge can be brought about.

EXAMPLE 6.1.1

For the object system O^1 of example 4.1, let X_i , $i = 1, 2, 3$ denotes the variable corresponding to the number in position i in the local state of O^i, and let the set of formulas $\{ X_i < X_j \mid i < j \}$ be the set of atomic formulas.

The formula corresponding to each equivalence class is given as follows:

$\{(1,2,3)\} := (X_1 < X_2) \wedge (X_1 < X_3) \wedge (X_2 < X_3)$

$\{(1,3,2), (3,1,2), (2,1,3)\} :=$

$\neg(X_1 < X_2) \wedge (X_2 < X_3) \vee (X_1 < X_2) \wedge (X_1 < X_3) \wedge \neg(X_2 < X_3)$ and

$\{(2,3,1), (3,2,1)\} := \neg (X_1 < X_3) \wedge \neg(X_2 < X_3)$.

If we denote the first formula by A_1 , the second by A_2 and the third by A_3 , then

$\Gamma(O^1) := \{ KA_1 \Rightarrow [a]\perp , KA_2 \Rightarrow \langle a \rangle KA_1 , KA_3 \Rightarrow \langle a \rangle KA_2 \}$, and

$PK(O^1) := \Gamma(O^1) \cup \{ KA_2 \Rightarrow \langle a \rangle KB \mid A_1 \Rightarrow B \} \cup \{ KA_3 \Rightarrow \langle a \rangle KC \mid A_2 \Rightarrow C \}$

$\cup \{ KA_2 \Rightarrow [a^2]\perp , KA_3 \Rightarrow \langle a^2 \rangle KA_1 , KA_3 \Rightarrow [a^3]\perp \}$

$\cup \{ KA_3 \Rightarrow \langle a^2 \rangle KD \mid A_1 \Rightarrow D \}$.

THEOREM 6.1.1

If O^i , O^j are behaviorally similar object systems, then there is an interpretation of O^i and an interpretation of O^j such that $PK(O^i) = PK(O^j)$.

PROOF

It follows from theorems 4.1.1 and 5.4.1 that there is an interpretation of O^i and an interpretation of O^j s.t. $\Gamma(O^i) = \Gamma(O^j)$. Hence $PK(O^i) = PK(O^j)$.

THEOREM 6.1.2

If object system O^i is an epimorphic image of object system O^j, then for any interpretation of O^i, there is an interpretation of O^j such that $PK(O^i) = PK(O^j)$.

PROOF:

It follows from theorem 5.4.2 that for any interpretation of O^i, there is an interpretation of O^j s.t. $\Gamma(O^i) = \Gamma(O^j)$. Hence $PK(O^i) = PK(O^j)$.

The following two corollaries follow from the above theorem and defintions 4.1.4 and 4.2.1 respectively.

COROLLARY 6.1.1

If object systems O^i and O^j belong to the same object class, then for any interpretation of O^i, there is an interpretation of O^j such that $PK(O^i) = PK(O^j)$ and vice versa.

COROLLARY 6.1.2

If object system O^j inherits from object system O^i, then for any interpretation of O^i, there is an interpretation of $O^j(i)$ such that $PK(O^j(i)) = PK(O^i)$.

6.2 RATIONAL BEHAVIOR

In [MH69], an agent's behavior is said to be rational if whenever it believes a sentence asserting that it should do something, it does it. In terms of our model, this is equivalent to saying that given an interpretation (Φ, V) of an object system O, the behavior of O is rational with respect to (Φ, V) if each knowledge transformation rule in PK(O) is true in the model $M = (O, \sim_Q, V)$.

In other words, the model $M = (O, \sim_Q, V)$ is a model of PK(O).

DEFINTION 6.2.1

Given an interpretation (Φ,V) of an object system O , the behavior of O is rational with respect to (Φ,V) if the model $M = (O, \sim_Q, V)$ is a model of PK(O).

For example, suppose O is an object system, and (Φ,V) is an interpretation of O s.t. the formulas $KA \Rightarrow <\alpha>KC$ and $K(A \wedge B) \Rightarrow [\alpha]\perp$ belong to PK(O). Then, the behavior of O is irrational with respect to (Φ,V) since whenever it knows A and B, it also knows A , and it is expected to be able to do at least the same things that it can do whenever it knows A. In this case, the model $M = (O, \sim_Q, V)$ is not a model of PK(O), since there is a local state of O (in the equivalence class corresponding to the formula $A \wedge B$) that satisfies the formula KA , but does not satisfy the formula $<\alpha>KC$.

LEMMA 6.2.1

The behavior of object system O is rational with respect to the interpretation (Φ,V) if and only if the model $M = (O, \sim_Q, V)$ is a model of $\Gamma(O)$.

PROOF

i) If M is a model of $\Gamma(O)$, then it is a model of PK(O) by the soundness of PLKA.

ii) If M is not a model of $\Gamma(O)$, then for some $t \in Q$ and $\bar{s} \in \bar{Q}$, $M,t \not\models K\bar{s} \Rightarrow \Gamma(\bar{s})$. Hence, $M,t \models K\bar{s}$ and $M,t \not\models \Gamma(\bar{s})$. Therefore, for some $\alpha = a \in A$ s.t. $R_a \neq \emptyset$ or $\alpha = (:e)$, $e \in E$ s.t. $S_e \neq \emptyset$, $M,t \models K\bar{s}$ and $M,t \not\models \Gamma(\bar{s},\alpha) \wedge \bar{\Gamma}(\bar{s},\alpha)$. And by definition of $\Gamma(\bar{s},\alpha) \wedge \bar{\Gamma}(\bar{s},\alpha)$, there is a formula B s.t. $M,t \not\models B$ and $K\bar{s} \Rightarrow B$ is a knowledge transformation rule logically implied by $\Gamma(O)$. Hence, there is a knowledge transformation rule $K\bar{s} \Rightarrow B$ in PK(O) s.t. $M,t \not\models K\bar{s} \Rightarrow B$.

THEOREM 6.2.1

The behavior of object system O is rational with respect to the interpretation (Φ,V), if and only if for any formulas $K\bar{s}_1 \Rightarrow \Gamma(\bar{s}_1)$ and $K\bar{s}_2 \Rightarrow \Gamma(\bar{s}_2)$ in $\Gamma(O)$, the formula $(K\bar{s}_1 \Rightarrow K\bar{s}_2) \Rightarrow (\Gamma(\bar{s}_1) \Rightarrow \Gamma(\bar{s}_2))$ is a theorem of PLKA.

PROOF

It follows from the definitions of the formulas $\bar{s}$ and $\Gamma(\bar{s})$ that $M,t \vdash K\bar{s}$ iff the formula $\bar{t} \Rightarrow \bar{s}$ is a tautology iff $\vdash K\bar{t} \Rightarrow K\bar{s}$ and $M,t \vdash \Gamma(\bar{s})$ iff $\vdash \Gamma(\bar{t}) \Rightarrow \Gamma(\bar{s})$.

a) If the behavior of O is rational with respect to (Φ,V), then from the above lemma, it follows that M is a model $\Gamma(O)$. Therefore, $\forall s_1, s_2 \in Q$, if $M,s_1 \vdash K\bar{s}_2$, then $M,s_1 \vdash \Gamma(\bar{s}_2)$. Hence, $\vdash K\bar{s}_1 \Rightarrow K\bar{s}_2$ implies $\vdash \Gamma(\bar{s}_1) \Rightarrow \Gamma(\bar{s}_2)$.

Therefore, $\vdash (K\bar{s}_1 \Rightarrow K\bar{s}_2) \Rightarrow (\Gamma(\bar{s}_1) \Rightarrow \Gamma(\bar{s}_2))$. From the completenes of PLKA, it follows that the formula $(K\bar{s}_1 \Rightarrow K\bar{s}_2) \Rightarrow (\Gamma(\bar{s}_1) \Rightarrow \Gamma(\bar{s}_2))$ is a theorem of PLKA.

On the other hand, if $(K\bar{s}_1 \Rightarrow K\bar{s}_2) \Rightarrow (\Gamma(\bar{s}_1) \Rightarrow \Gamma(\bar{s}_2))$ is a theorem of PLKA, then by soundness of PLKA, $\vdash (K\bar{s}_1 \Rightarrow K\bar{s}_2) \Rightarrow (\Gamma(\bar{s}_1) \Rightarrow \Gamma(\bar{s}_2))$.

Then $M,s_1 \vdash (K\bar{s}_1 \Rightarrow K\bar{s}_2) \Rightarrow (\Gamma(\bar{s}_1) \Rightarrow \Gamma(\bar{s}_2))$. Therefore, $M,s_1 \vdash (K\bar{s}_1 \Rightarrow K\bar{s}_2)$ implies $M,s_1 \Rightarrow (\Gamma(\bar{s}_1) \Rightarrow \Gamma(\bar{s}_2))$. Hence, $M,s_1 \vdash K\bar{s}_2$ implies $M,s_1 \Rightarrow \Gamma(\bar{s}_2)$.

So, M is a model of $\Gamma(O)$; and from the above lemma, it follows that the behavior of O is rational.

7 KNOWLEDGE AND COMPUTATION

As discussed in section 5, an agent's knowledge may be characterized as static or dynamic. Dynamic knowledge refers to what we learn or are aware of as a result of performing some activities or interacting with other agents, and obviously may "forget" for the same reasons. On the other hand, static or local knowledge refers to what we know based uniquely on our perception of the world, and may cease to know only if we change this perception of the world. We cease to know what we knew because we no longer have the evidence that it is true, and we forget what we have learned (or have been aware of) because of the contradiction brought by new events.

It is evident that these two aspects of knowledge are related since the performance of activities and interactions with other agents (which by the way may change our knowledge state) depend on what we already knew (i.e. our previous knowledge state). In other words, we "update" our knowledge by learning, and we learn by performing some activities or interacting with other agents, which depend on what we already knew. However, there are situations where it might not be possible for an observer to determine the relevance of an agent's local knowledge with respect to the performance of its activities or interactions with other agents. For instance, suppose you are the instructor of a course and there is a student in your class that you find particularly brilliant. However, (s)he did not perform well on class tests (given the tests criteria). My first question is to ask what grade you will give to this student. Secondly, are you going to have some doubts about his/her brilliance? Assuming that you do not, the question now is to know whether or not the tests were consistent with that student's local knowledge. This brings us to the heuristic part of intelligence of J. McCarthy and P.J. Hayes [MH69]. That means the mechanism that on the basis of the information solves the problem and decides what to do. Although an object's protocol (i.e. what it can do at any local state) depends on its local or static knowledge, what an object instance actually chooses to do at any step of the system computation depends on: Its local knowledge that it is aware of; what has happened in the system so far and that it is aware of;

what it has learned so far; its goal, and its ability to evaluate the known consequences of its actions.

For example, in a two players game, if a player is aware that the other will attack, he may want to attack first if by doing so he will gain some advantage. I may also want to get into a fight with someone who has hurt my feelings. However, if I am aware of his strength or that I will lose the fight, I may decide not to do so.

In the following, we discuss some issues about learning and awareness in distributed object-based systems. The syntax and the semantics of a language for talking about learning/awareness in distributed object-based sytems are introduced in section 7.2 and in section 7.3, we prove some properties of learning/awareness and forgetting. Note that our approach is closely related to the work of K.M. Chandy and J. Misra [CM86], and that of R. Parikh and R. Ramanujam [PR85].

From now on we assume given for each object O^i in the DOBS an object instance (O^i,q^i_0).

7.1 LEARNING AND AWARENESS IN A DOBS

We propose a language for the specification and reasoning about learning/awareness in distributed object-based systems. The language $\mathcal{L}$ is built from a countable set Φ^0 of atomic formulas, and the languages $\mathcal{L}^i(\Phi^i,A,E)$ (one for each object) defined as in section 5. Note that this approach to the language $\mathcal{L}$ is similar to that suggested in [PR85]. The set of atomic formulas Φ^0 corresponds to basic facts about the DOBS that are not described in objects' task-domain situations. That means the properties of computations. It is assumed that $\Phi^i \cap \Phi^j = \emptyset$ for $i \neq j$. For each object O^i, the language $\mathcal{L}^i$ is used to describe and reason about its local/static knowledge. The knowledge operator K in each language $\mathcal{L}^i$ is now indexed by i (i.e. K_i). From each language $\mathcal{L}^i$, only the formulas of the form K_iA where A is a K-free formula are included in $\mathcal{L}$. The intuition is that only the facts about an object task-domain situations that it knows are relevant to system computations.

Although a formula K_iA (where A is a K-free formula) still has its meaning in language $\mathcal{L}^i$, it will carry on another meaning in the language $\mathcal{L}$.

In the language $\mathcal{L}$, the formula K_iA expresses the fact that object instance (O^i,q^i_0) is sure about its knowledge of A. Note that with respect to computations, the problem is not limited to whether or not an object instance knows a fact about its task-domain situations. We must also be able to say whether or not that knowledge may be useful for the next step in the computation. For instance, suppose your neighbor calls you and asks if you have a pen. You have a writting pen, but not a play pen. Unless you answer "yes" or "no", it would not be possible for an observer to determine which type of pen you have in mind after the call. In this circumstance, we will say that you are unsure about your knowledge of having the pen, whereas your neighbor's question is inconsistent with your knowledge of having a pen. Of course, things would have been different if you had the two types of pen. Note also that in case you do not have either type of pen, your answer is not determined by your knowledge of having the pen, but instead by your knowledge of not having the pen. In this example, to decide whether to say "yes" or "no", you surely must have taken into consideration some facts about your past interactions with your neighbor that you have learned or are aware of or simply, you have taken a lucky guess. So, object instance (O^i,q^i_0) will be said to be sure about its knowledge of A (K_iA) (where A is a K-free formula in $\mathcal{L}^i$) with respect to system computation x if and only if for each local state s reachable by (O^i,q^i_0) in computation x (i.e. $s \in T^i(x)$) O^i knows A at s in the language $\mathcal{L}^i$. In other words, it is sure about its local knowledge in the system computation x if and only if it has no doubts about it after computation x. The knowledge state of an object instance (O^i,q^i_0) after a computation x will correspond to all of its local knowledge that it is sure about after computation x (i.e. the formulas K_iA true at every $s \in T^i(x)$). Note that our notion of "sure" is different from that of [CM86].

The language $\mathcal{L}$ also has a new operator L_i (one for each object instance) to express learning or awareness. The formula L_iA will be read (O^i,q^i_0) is aware of A, whereas the formula L_iK_iA is best read (O^i,q^i_0) is aware of its knowledge of A.

Object instance (O^i,q^i_0) will be said to be aware of fact A (L_iA) in system computation x if and only if A is true in all system computations that it cannot distinguish from x (i.e. i-isomorphic to x).

Learning via a string of objects is denoted by the formula $L<i_1 i_2 \ldots i_k>A$ (where the i_j's need not be all distinct) which expresses the fact that $(O^{i},q^{i'}_0)$ learns from $(O^{i},q^{i''}_0)$, which learns from ... , (O^{i},q^{i*}_0) that A. The formula $L<i_1 i_2 \ldots i_k>A$ is satisfied in system computation x if for every sequence of system computations $y_0=x$, y_1 , ... , y_k s.t. y_{j-1} is i_j-isomorphic to y_j , y_k satisfies A. In other words, as far as $(O^{i},q^{i'}_0)$ is concerned in computation x , A could not have been otherwise for other object instances. Note that this formula can also be seen as an abbreviation of the formula $Li_1(Li_2(\ldots (Li_k(A) \ldots))$. Also, observe that this aspect of learning/awareness corresponds to the intuitive notion of learning by experience. This contrasts with another form of learning which corresponds to the transfer of information between agents, with resulting effect the change of the receiving agent perception of the world (knowledge partition). In this respect, the agent's knowledge is not identified with the information received as it is done in [Mos92]. We postpone further discussions on this aspect of learning to a later time.

In the following, I denotes the set $\{1,2,\ldots,n\}$ where n is the number of objects in the DOBS. The syntax and the semantics of the language $\mathcal{L}$ are defined as follows:

7.2 SYNTAX AND SEMANTICS OF THE LANGUAGE $\mathcal{L}$

Given a countable set of formulas Φ^0 and the lanuages $\mathcal{L}^i(\Phi^i, A, E)$ $(i \leq n)$ defined as in section 5 s.t. $\Phi^i \cap \Phi^j = \emptyset$ for $i \neq j$, we define the language $\mathcal{L}$ as follows:

$\top \in \mathcal{L}$ and

if $p \in \Phi^0$ then $p \in \mathcal{L}$

$\forall i \in I$, if $A \in \mathcal{L}^i$ (where A is a K-free formula) then $K_iA \in \mathcal{L}$

if $A , B \in \mathcal{L}$ and $i \in I$, then $\neg A$, $A \wedge B$ and L_iA are in $\mathcal{L}$

We also use the following syntactic abbreviations where $A , B \in \mathcal{L}$: $\bot := \neg \top$; $A \vee B := \neg(\neg A \wedge \neg B)$; $A \Rightarrow B := \neg A \vee B$; and $A \Leftrightarrow B := (A \Rightarrow B) \wedge (B \Rightarrow A)$.
Also, if $i_1, i_2, \ldots, i_k \in I$, $k \geq 2$ then $L<i_1 i_2 \ldots i_k>A := Li_1(Li_2(\ldots (Li_k(A) \ldots))$.
The formula K_iA says that agent i is sure about its knowledge of A, and L_iA says that agent i is aware of A (L_iK_iA is best read agent i is aware of its knowledge of A),

and $L<i_1 i_2 \ldots i_k>A$ says that agent i_1 learns from agent i_2 which learns from ... agent i_k that A.

The semantics of $\mathcal{L}$ is defined with respect to the set $SC(O, q^1_0, \ldots, q^i_n)$ of system computations, a model M^i for each language $\mathcal{L}^i$ and an interpretation of the atomic formulas Φ^0.

An interpretation of the atomic formulas is a function $V : \Phi^0 \dashrightarrow \mathcal{P}(SC(O, q^1_0, \ldots, q^n_0))$. We will also require as in [CM86] for each $p \in \Phi^0$ and system computations x and y that $x \in V(p)$ iff $y \in V(p)$ if for each $i \in I$, $x^i = y^i$.

Given an interpretation V of the atomic formulas, a model M^i for each language $\mathcal{L}^i$ and a system computation x, the satisfaction relation $x \vdash A$ is defined by induction on the structure of $A \in \mathcal{L}$ as follows:

$x \vdash T$ and

if $A \in \Phi^0$ then $x \vdash A$ iff $x \in V(A)$.

$x \vdash \neg A$ iff not $x \vdash A$

$x \vdash A \wedge B$ iff $x \vdash A$ and $x \vdash B$

$x \vdash K_i A$ iff $\forall q \in T^i(x)$, $M^i, q \vdash_i K_i A$ (where $\vdash_i$ is the satisfaction relation for the language $\mathcal{L}^i$)

$x \vdash L_i A$ iff $\forall y$ s.t. $x \sim_i y$, $y \vdash A$

Given an interpretation V of the atomic formulas and a model M^i for each language $\mathcal{L}^i$, a formula A is true if $\forall x \in SC(O, q^1_0, \ldots, q^n_0)$, $x \vdash A$.

It is valid (denoted by $\vdash A$) if it is true for every interpretation V of the atomic formulas and model M^i for each language $\mathcal{L}^i$.

THEOREM 7.2.1

The following formula schemata are valid:

a) Local/Static Knowledge:

1) $K_i T$

2) $\neg K_i A \vee \neg K_i \neg A$

3) $K_i A \wedge K_i B \Leftrightarrow K_i(A \wedge B)$

4) $K_i A \vee K_i B \Rightarrow K_i(A \vee B)$

5) $K_i A \wedge K_i(A \Rightarrow B) \Rightarrow K_i B$

b) Awareness:

1) $L_i A \Rightarrow A$

2) $L_i A \Rightarrow L_i L_i A$

3) $L_i A \wedge L_i(A \Rightarrow B) \Rightarrow L_i B$

4) $\neg L_i A \Rightarrow L_i(\neg L_i A)$

5) $L_i K_i A \vee L_i \neg K_i A$

c)

1) $K_i A \Leftrightarrow L_i K_i A$

2) $\neg K_i A \Leftrightarrow L_i \neg K_i A$

3) $L_i K_j A \Leftrightarrow L_{<ij>} K_j A$

4) $L_i \neg L_j A \Rightarrow \neg L_{<ij>} A$

5) $L<i_1 \ldots i_k> A \Rightarrow L<i_j \ldots i_k> A \quad \forall j \geq 2$

6) $L<i_1 \ldots i_k> A \Rightarrow L i_j A \quad \forall j \geq 1$

PROOF

Formula schemata a)1 to a)5 are valid in the language $\mathcal{L}^i$. Therefore, they are valid in the language $\mathcal{L}$. b)1 to b)4 correspond to the axioms of the logic S5 of knowledge and their proofs are similar to those of these axioms. b)5 is easily derived from the semantics of L_i and K_i. c)1 to c)6 are derived from b)1 to b)5) using modus ponens.

It can also be shown easily that the valid formulas are closed under the generalization of L_i ($i \geq 1$) rule of inference. Note that b)5 corresponds to the "sure" condition of [CM86]; and since it is valid, it follows that the formulas K_iA ($i \geq 1$) correspond to "local predicates" of [CM86].

7.3 LEARNING/AWARENESS AND FORGETTING CONDITIONS

In the following, we prove some properties of learning/awareness and forgetting in a DOBS. Some of these results can be seen as an improvement on those obtained in [CM86].

For simplicity, we assume that the DOBS contains only two objects (i.e. $I = \{1,2\}$). x is a system computation and y is a sequence of events such that xy is a system computation.

LEMMA 7.3.1 (Learning/Awareness Conditions)

For any formula A, if $x \not\vdash L_{<ij>}A$ and $xy \vdash L_{<ij>}A$ $i \neq j$,

then (O^i, q^i_0) has a receive i-event in y. Furthermore, if $y = y_1R(i,.,.,j)y_2$ then:

i) $x \not\sim_i xy_1R(i,.,.,j)$

ii) For every sequence of events y' s.t. $xy_1R(i,.,.,j) \preceq xy' \preceq xy$, $xy' \vdash L_jA$

Note that in the similar result obtained in [CM86] it is required that $x \not\vdash L_jA$, which is stronger than our condition. In fact, their result is obtained as a special case of ours. Also, our result does not require that the corresponding send event occurs in y.

PROOF

a) $x \not\vdash L_{<ij>}A$ iff $\exists z_1, z_2$ s.t. $x \sim_i z_1$, $z_1 \sim_j z_2$ and $z_2 \not\vdash A$.

If there is no receive i-event in y, then by proposition 3.5.2 a)i), xy^i is a system computation and $xy^i \sim_i xy$. By proposition 3.5.2 c)ii), z_1y^i is a system computation and $z_1y^i \sim_i xy^i$. And by proposition 3.5.2 b), $z_1y^i \sim_j z_1$.

Hence, $xy \sim_i z_1y^i \sim_j z_2$. Therefore, $z_2 \vdash A$ since $xy \vdash L_{<ij>}A$. (Contradiction).

b) Let $y = y_1R(i,.,.,j)y_2$.

To prove the remaining part of the lemma, we will assume (without lost of generality) that there is no receive i-event in y_2 (otherwise take the last one).

i) By proposition 3.5.2ai) , $xy_1R(i,.,.,j)y'_2$ is a system computation and $xy_1R(i,.,.,j)y'_2 \sim_i xy_1R(i,.,.,j)y_2 = xy$. If $x \sim_i xy_1R(i,.,.,j)$, then $z_1 \sim_i xy_1R(i,.,.,j)$ and by proposition 3.5.2 c)i) & ii), $z_1y'_2$ is a system computation and $z_1y'_2 \sim_i xy_1R(i,.,.,j)y'_2 \sim_i xy$. But $z_1y'_2 \sim_j z_1$ by proposition 3.5.2 b).

Hence, $xy \sim_i z_1y'_2 \sim_j z_2$. Therefore $z_2 \vdash A$. (Contradiction).

ii) Let y' s.t. $xy_1R(i,.,.,j) \preceq xy' \preceq xy$ and $xy = xy'y'_2$.

Since there is no receive i-event in y_2 , there is no receive i-event in y'_2.

Also, if $xy' \not\vdash L_jA$, then $xy' \not\vdash L_{<ij>}A$. But since $xy'y'_2 \vdash L_{<ij>}A$, there is a receive i-event in y'_2 by the first part of the lemma. (Contradiction).

COROLLARY 7.3.1

For any formula A , if $x \not\vdash L_iA$ and $xy \vdash L_{<ij>}A$ $i \neq j$,

then (O^i,q^i_0) has a receive i-event in y. Furthermore , if $y = y_1R(i,.,.,j)y_2$ then:

i) $x \not\sim_i xy_1R(i,.,.,j)$

ii) For every sequence of events y' s.t. $xy_1R(i,.,.,j) \preceq xy' \preceq xy$, $xy' \vdash L_jA$

REMARK: An agent i may learn a fact A from another agent j during a computation, only if it receives from j a message that changes its knowledge state. Moreover, the message must be received after agent j becomes aware of A.

PROOF If $x \not\vdash L_iA$, then $x \not\vdash L_{<ij>}A$. The proof then follows from Lemma 7.3.1.

COROLLARY 7.3.2

For any formulas K_jA and $K_j\neg A$, if $x \vdash K_j\neg A$ and $xy \vdash L_iK_jA$ $i \neq j$,

then (O^i,q^i_0) has a receive i-event in y. Furthermore , if $y = y_1R(i,.,.,j)y_2$ then:

i) $x \not\sim_i xy_1R(i,.,.,j)$

ii) For every sequence of events y' s.t. $xy_1R(i,.,.,j) \preceq xy' \preceq xy$, $xy' \vdash K_jA$.

REMARK: If agent j is sure about its knowledge of $\neg A$ and then agent i becomes aware that it is sure about its knowledge of A , then i must have received a message from j that changed its knowledge state. Moreover, the message must have been received after j becomes sure about its knowledge of A.

PROOF

If $x \vdash K_j \neg A$ then $x \nvdash L_i K_j A$, and $xy \vdash L_i K_j A$ iff $xy \vdash L_{<ij>} K_j A$.
The proof then follows from Corollary 7.3.1.

Note that the same conclusion applies for the case $x \nvdash KA$

THEOREM 7.3.1 (Object failure detection)

For any formula $K_j A$, there is no system computation z such that
$\forall z' \prec z$, $z' \nvdash K_j A$ and $z \vdash L_i K_j A$, $i \neq j$.
In particular, object instance failure detection is impossible without message delay.

PROOF

a) Let's assume that there is a system computation $z = xy$ s.t. $\forall z' \prec z$, $z' \nvdash K_j A$ and $z \vdash L_i K_j A$. Therefore, $x \nvdash K_j A$ and since $z \vdash L_i K_j A$ it follows from the above corollary that there is a receive i-event in y and that if $z = xy_1 R(i, \ldots, j) y_2$, then $xy_1 R(i, \ldots, j) \vdash K_j A$. But by hypothesis, $xy_1 R(i, \ldots, j) \nvdash K_j A$ (contradiction).
b) We will say that an object has failed at a local state q if and only if it cannot perform any internal or send event at q. But it can still be resumed by a receive event at that state (i.e. failure may not persist). Note that an object fails at a local state if and only if it knows it.
If A is the proposition representing the failure of O^j , then (O^j, q^j_0) fails after computation z iff $z \vdash K_j A$ and this failure is detected by (O^j, q^i_0) iff $z \vdash L_i K_j A$.
If we assume that initially the failure did not hold, then $z = xy$ s.t. $x \nvdash K_j A$ and $xy \vdash L_i K_j A$. It then follows from the above corollary that there is a receive i-event in y s.t. if $y = y_1 R(i, \ldots, j) y_2$ then $xy_1 \vdash K_j A$.

Therefore, for any local state $q \in T^j(xy_1)$, O^j cannot perform a send event at q. Hence, (O^j,q^j_0) must have performed the corresponding send event before it performs the event that took it to the fail state. Otherwise, failure detection is impossible.

LEMMA 7.3.2 (Forgetting Condition)

For any formula A , if $x \models L_{<ij>}A$ and $xy \not\models L_jA$ $i \neq j$

then (O^j,q^j_0) has a receive j-event in y. Furthermore, the corresponding send i-event is in y such that if $y = y_1S(i,.,.,j)y_2R(j,.,.,i)y_3$, then:

i) $x \not\sim_i xy_1S(i,.,.,j)$

ii) $x \not\sim_j xy_1S(i,.,.,j)y_2R(j,.,.,i)$

REMARK: An agent may forget what another agent has learned from it only if it receives from that agent a message that changes its knowledge state. Moreover, the message must be sent after the second agent has already changed its own knowledge state.

PROOF

a) If there is no receive j-event in y , then xy^j is a system computation and $xy^j \sim_j xy$ (by proposition 3.5.2 a)i)). Therefore, $xy^j \not\models L_jA$ since $xy \not\models L_jA$. But $xy^j \sim_i x$ by proposition 3.5.2b). Then $xy^j \models L_jA$. (Contradiction).

If every receive j-event in y does not have its corresponding send i-event in y , then xy^j is a system computation and $xy^j \sim_j xy$ by proposition 3.5.2a)ii). So, we have the same condition as above, that results to a contradiction.

b) Let $y_4 = y_2R(j,.,.,i)y_3$. To prove the remaining part of the lemma, we assume (without lost of generality) that there is no send i-event in y_4 with corresponding receive j-event in y_4 (otherwise take the last one).

i) So , $xy_1S(i,.,.,j)y^j_4$ is a system computation and $xy_1S(i,.,.,j)y^j_4 \sim_j xy$ by proposition 3.5.2a)ii). Then $xy_1S(i,.,.,j)y^j_4 \not\models L_jA$. But , $xy_1S(i,.,.,j) \sim_i xy_1S(i,.,.,j)y^j_4$ by proposition 3.5.2b). Therefore, if we assume that $x \sim_i xy_1S(i,.,.,j)$, then $xy_1S(i,.,.,j)y^j_4 \sim_i x$. Hence $xy_1S(i,.,.,j)y^j_4 \models L_jA$. (Contradiction).

ii) Since there is no receive j-event in y_3 with corresponding send i-event in y_4, $xy_1S(i,.,.,j)y_2R(j,.,.,i)y'_3$ is a system computation which is j-isomorphic to xy by proposition 3.5.2a)ii). If we assume that $x \sim_j xy_1S(i,.,.,j)y_2R(j,.,.,i)$, then by proposition 3.5.2c) xy'_3 is a system computation and
$xy'_3 \sim_j xy_1S(i,.,.,j)y_2R(j,.,.,i)y'_3 \sim_j xy$. Hence, $xy'_3 \not\vdash L_jA$.
But, since $xy'_3 \sim_i x$, $xy'_3 \vdash L_jA$. (Contradiction).

COROLLARY 7.3.3

For any formula K_jA, if $x \vdash L_iK_jA$ and $xy \not\vdash K_jA$ $i \neq j$
then (O^j,q^j_0) has a receive j-event in y. Furthermore, the corresponding send i-event is in y such that if $y = y_1S(i,.,.,j)y_2R(j,.,.,i)y_3$ then:

i) $x \not\vdash_i xy_1S(i,.,.,j)$

ii) $x \not\vdash_j xy_1S(i,.,.,j)y_2R(j,.,.,i)$

REMARK: An agent may become unsure about its local knowledge that another agent has been aware of only if it receives from that agent a message that changes its knowledge state. Moreover, the message must be sent after the second agent has already changed its own knowledge state.

PROOF

$x \vdash L_iK_jA$ iff $x \vdash L_{<ij>}K_jA$ and $xy \not\vdash K_jA$ iff $xy \not\vdash L_jK_jA$.
The proof then follows from Lemma 7.3.2

COROLLARY 7.3.4

For any formulas K_jA and $K_j\neg A$, if $x \vdash L_iK_jA$ and $xy \vdash K_j\neg A$ $i \neq j$
then (O^j,q^j_0) has a receive j-event in y. Furthermore, the corresponding send i-event is in y such that if $y = y_1S(i,.,.,j)y_2R(j,.,.,i)y_3$ then:

i) $x \not\vdash xy_1S(i,.,.,j)$

ii) $x \not\vdash xy_1S(i,.,.,j)y_2R(j,.,.,i)$

PROOF

If $xy \models K_j \neg A$ then $xy \not\models K_j A$. The proof then follows from the corollary 7.3.3

THEOREM 7.3.2

For any formula A , if $x \models L_{<ij>} A$ and $xy \models L_{<ij>} \neg A$ $i \neq j$

then (O^i, q^i_0) has a receive i-event in y. Furthermore, if $y = y_1 R(i,.,.,j) y_2$ then:

a) For every sequence of events y' such that $xy_1 R(i,.,.,j) \preceq xy' \preceq xy$, $xy' \models L_j \neg A$

b) (O^j, q^j_0) has a receive j-event in y_1 with the corresponding send i-event in y_1 such that if $y_1 = y_3 S(i,.,.,j) y_4 R(j,.,.,i) y_5$ then:

   i) $x \not\vdash_i xy_3 S(i,.,.,j)$

   ii) $x \not\vdash_j xy_3 S(i,.,.,j) y_4 R(j,.,.,i)$

PROOF

$x \models L<ij>A$ implies $x \not\models L<ij> \neg A$. It then follows from lemma 7.3.1 that (O^i, q^i_0) has a receive i-event in y and that if $y = y_1 R(i, \ldots, j) y_2$ then for every sequence of events y' s.t. $xy_1 R(i,.,.,j) \preceq xy' \preceq xy$, $xy' \models L_j \neg A$. Therefore , $xy_1 \models L_j \neg A$. Hence $xy_1 \not\models L_j A$, and from lemma 7.3.2 it follows that (O^j, q^j_0) has a receive j-event in y_1 with the corresponding send i-event in y_1 s.t. if $y_1 = y_3 S(i,.,.,j) y_4 R(j,.,.,i) y_5$ then $x \not\vdash_i xy_3 S(i,.,.,j)$ and $x \not\vdash_j xy_3 S(i,.,.,j) y_4 R(j,.,.,i)$.

REFERENCES

[Ab87] S. Abramsky, "Observation Equivalence as a Testing Equivalence", Theoretical Computer Science 58 (1987) pp 225-241.

[Ac88] P. Aczel, "Non-Well Founded Sets", CSLI Lecture Notes no. 14 (1988).

[An91] G. R. Andrews, "Paradigms for Process Interaction in Distributed Programs", in ACM Computing Surveys, vol. 23, no. 1, March 91, pp 49-90.

[AV90] P. America and F. Van der Linden, "A Parallel Object-Oriented Language with Inheritance and Subtyping", OOPSLA ECOOP 1990 Proceedings, ed. N. Meyrowitz, PP 161-168.

[B77] M. Bunge, Treatise on Basic Philosophy: Vol. 3: Ontology I: The Future of the World, Reidel, Boston, 1977.

[B79] M. Bunge, Treatise on Basic Philosophy: Vol. 4: Ontology II: A World of Systems, Reidel, Boston, 1979.

[BACH90] S. Bear, P. Allen, D. Coleman, and F. Hayes, "Graphical Specification of Object Oriented Systems", OOPSLA ECOOP 1990 Proceedings, ed. N. Meyrowitz, pp 28-31.

[BHR84] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe, "A Theory of Communicating Sequential Processes", JACM, vol. 31 (1984) pp 560-569.

[CC91] R.S. Chin and S.T. Chanson, "Distributed Object-Based Programming Systems", ACM Computing Surveys, vol. 23, no. 1, March 91, pp 91-124.

[CM86] K.M. Chandy and J. Misra, "How Processes Learn", Distributed Computing, Springer-Verlag 1986, vol. 1, pp 40-52.

[D81] F. Dretske, Knowledge and the Flow of Information, MIT press 1981.

[EG89] C.A. Ellis and S.J. Gibbs, "Active Objects: Realities and Possibilities", in Object-Oriented Concepts, Database, and Applications, ed. W. Kim and F. Lochovsky, Addison-Wesley, Reading, MA, 1989, pp 561-572.

[EGR91] C.A. Ellis, S. J. Gibbs, and G. L. Rein, "Groupware: Some Issues and Experineces", Communications of the ACM, Vol. 34, Number 1, January 1991, PP 39-58.

[FL76] M. Fischer and R. Ladner, "Propositional Modal Logics of Programs", 9th ACM STOC symposium (1976) pp 286-294.

[G87] R. Goldblatt, Logics of Time and Computation, CSLI Lecture Notes no. 7 (1987).

[GN87] M.R. Genesereth and N.J. Nilsson, Logical Foundations of Artificial Intelligence, Morgan Kaufmann publisher, inc., (1987).

[H87] D. Harel, "Statecharts: A Visual Formalism for Complex Systems", Science of Computer Programming 8, (1987) pp 231-274.

[HF89] J.Y. Halpern and R. Fagin, "Modelling Knowledge and Action in Distributed Systems", Distributed Computing, Springer-Verlag 1989, vol. 3, pp 159-177.

[HHG90] R. Helm, I.M. Holland and D. Gangopadhyay, "Contracts: Specifying Behavioral Compositions in Object-Oriented Systems", OOPSLA ECOOP 1990 Proceedings, ed. N. Meyrowitz, pp 28-31.

[HM85] M. Hennesy and R. Milner, "Algebraic Laws for Nondeterminism and Concurrency", JACM, vol. 32, no. 1, Jan. 1985, pp 137-161.

[HPSS87] D. Harel, A. Pnueli, J.P. Schmidt and R. Sherman, "On the Formal Semantics of Statecharts", Proceedings of the second IEEE symposium on logic in computer science, (1987) pp 54-64.

[K92] W. Kent, "User Object Models", OOPS Messenger, vol. 3, no. 1, Jan. 1992, pp 10-25.

[KP81] D. Kozen and R. Parikh, "An Elementary Proof of the Completeness of PDL", theor. comp. sci., 14 (1981) pp 113-118.

[KPN90] P. Krasucki, R. Parikh and G. Ndjatou, "Probabilistic Knowledge and Probabilistic Common Knowledge", proceedings of the fifth international symposium on methodologies for intelligent systems, (1990), pp 1-8.

[KT89] D. Kozen and J. Tiuryn, "Logics of Program", technical report 89-962, Dept. of Computer Science, Cornell University, Ithaca, NY, Jan. 89.

[L89] L. Lamport, "A Simple Approach to Specifying Concurrent Systems", Communications of the ACM, Jan. 1989, vol. 32, no. 1, pp 32-45.

[M80] R. Milner, "A Calculus of Communicating Systems", Lecture Notes in Computer Science, vol. 92, Springer-Verlag, (1980).

[Mo] R.C. Moore, "Reasoning about Knowledge and Action", in J. Hobbs and R.C. Moore (Ed.), Formal Theories of the Commonsense World, Ablex 1985, pp 319-358.

[Mos92] Y. Moses, "Knowledge and Communication (A Tutorial)", Proceedings of the Fourth Conference on Theoretical Aspects of Reasoning about Knowledge, ed. Y. Moses, 1992, pp 1-14.

[MH69] J. McCarthy and P.J. Hayes, "Some Philosophical Problems from the StandPoint of Artificial Intelligence", in B. Meltzer and D. Michie (eds.) Machine Intelligence 4, pp 463-502, Edinburgh University Press.

[Ne88] G. Neiger, "Knowledge Consistency: A Useful Suspension of Disbelief (Preliminary Report)", Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge, ed. M.Y. Vardi, 1988, pp 295-308.

[Ni89] O. Nierstrasz, "A Survey of Object-Oriented Concepts", in Object-Oriented Concepts, Databases and Applications, ed. W. Kim and F. Lochovsky, Addison-Wesley, Reading, MA, 1989, pp 3-21.

[NP90] O.M. Nierstrasz and M. Papathomas, "Viewing Objects as Patterns of Communicating Agents", OOPSLA ECOOP 1990 Proceedings, ed. N. Meyrowitz, pp 38-43.

[NPW81] M. Nielsen, G. Plotkin and G. Winskel, "Petri Nets, Event Structures and Domains, Part I", theor. comp. sci. 13 (1981) pp 85-108, North-Holland Publishing Co.

[NT89] O. Nierstrasz and D. C. Tsichritzis, "Integrated Office SYstems", in Object-Oriented Concepts, Database, and Applications, ed. W. Kim and F. Lochovsky, Addison-Wesley, Reading, MA, 1989, pp 199-215.

[Pa92] R. Parikh, "Utilities and the Semantics of Common Nouns", presented at the 3rd meeting on mathematics of languages, Austin, Texas, November 92.

[Pa90] R. Parikh, "Recent Issues in Reasoning about Knowledge", Proceedings of the Third Conference (TARK 1990), ed. R. Parikh, pp 3-9.

[Pa87] R. Parikh, "Knowledge and the Problem of Logical Omniscience", Methodologies for Intelligent Systems Proceedings, ed. Z.W. Ras and M. Zemankova, pp 432-439.

[Pn85] A. Pnueli, "Linear and Branching Structures in the Semantics and Logics of Reactive Systems", 12th ICALP, LNCS 194, 1985.

[PR85] R. Parikh and R. Ramanujam, "Distributed Processes and the Logic of Knowledge", in Lecture Notes in Computer Science, vol. 193, Springer-Verlag, (1985), pp 256-268.

[RWZ92] N. Reingold, D.W. Wang and L.D. Zuck, "Games I/O Automata Play", lecture notes in computer science 630, W.R Cleaveland (Ed.), Springer-verlag, pp 325-339

[TS89] C. Tomlinson and M. Scheevel, "Concurrent Object-Oriented Programming Languages", in Object-Oriented Concepts, Database, and Applications, ed. W. Kim and F. Lochovsky, Addison-Wesley, Reading, MA, 1989, pp 79-124.

[W49] W. Weaver, "Recent Contributions To The Mathematical Theory Of Communication", in The Mathematical Theory Of Communication, by C. Shannon and W. Weaver, The University of Illinois Press: URBANA, 1949.

[Wa89] Y. Wand, "A Proposal For A Formal Model Of Objects", in Object-Oriented Concepts, Database, and Applications, ed. W. Kim and F. Lochovsky, Addison-Wesley, Reading, MA, 1989, pp 537-559.