Order Number 9119642

Knowledge formalization and representation of the dynamics of expressive musical performance

Johnson, Margaret Louise, Ph.D.

City University of New York, 1991

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

Knowledge Formalization and Representation of the Dynamics of Expressive Musical Performance

by

Margaret Johnson

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.

1991

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.

1/9/91 — Date

Chair of Examining Committee

1/9/91 — Date

Executive Officer

Professor George Klir
Professor Miriam Tausner
Professor Thomas Wesselkamper

Supervisory Committee

The City University of New York

ABSTRACT

Knowledge Formalization and Representation of the Dynamics of Expressive Musical Performance

by

Margaret Johnson

Advisor: Professor Robert Orchard

An expert system that determines interpretations of Bach fugues is developed. The input to the system is a numeric representation of the fugue, and the output is a listing of performance actions. Were a musician to perform these actions while playing the fugue, the performance would sound like one given by the expert upon whose expertise the system is based. A program called Music Printer Plus by Temporal Acuity Products is used to actually generate MIDI (Musical Instrument Digital Interface) performances of the fugue before and after the knowledge base is applied.

We describe the process of eliciting, formalizing and representing the knowledge of two expert performers. The systems theory concept of knowledge structures (as developed by George Klir), is used to organize the elicited knowledge, and as a basis for the design of the system. The actual implementation is based on a language recognizer for rhythmic patterns. Once a particular pattern is recognized in the input, a

specific rule is applied, or a procedure is performed.

The expert system is run on example fugues to demonstrate the entire process. We also discuss the implementation issues involved in developing "human-sounding" performances.

ACKNOWLEDGEMENTS

I wish to acknowledge my debt to several people who made this dissertation project possible. My advisor, Robert Orchard, provided valuable suggestions, ideas and criticisms. All the members of my committee: George Klir, Miriam Tausner, Thomas Wesselkamper, and Professor Orchard, were very supportive and enthusiastic about the project, and provided lots of helpful suggestions. I wish to give a special thanks to George Klir, without whose research none of this would have been possible.

During the past year, I received a great deal of support from my friends and associates at Integrity Life: Barbara Taffet, Mary Ann O'Brien, Dale Hesser and Laurie Salo. They all provided interesting "diversions" for me at a time when I needed them most.

I also wish to thank Shirley S. Levene and Florence Radin without whom I can honestly say, none of this would have been possible. To Judy Barrett and Susan Johnson: you two owe me some DP next Christmas in San Francisco.

Finally, to my husband David, thank you for your infinite patience and support. It is finally over; now, we can settle down to a calm and peaceful life...

CONTENTS

List of Figures

1. Introduction

Expert systems are computer systems developed to solve complex, real-world problems by emulating the problem-solving processes of human experts. The earliest expert systems were created in the mid-60's with the DENDRAL project at Stanford which performed organic chemical analysis (Feigenbaum, Buchanan, and Lederberg); and, the MACSYMA project at M.I.T., which performed symbolic integration and formula simplification (Martin and Moses).

Over the past thirty years, expert systems have been developed in many domains, the following being a representative list:

a) medicine: diagnostic systems for the treatment of various diseases
   - MYCIN for infectious disease diagnosis (Shortliffe, 1976);
   - CASNET for glaucoma diagnosis (Weiss, Kulikowski, and Safir, 1977);
   - CADUCEUS for internal disease diagnosis (Pople, 1977);

b) education: interactive tutoring systems
   - SCHOLAR, a tutor in geography (Carbonell, 1970);

- SOPHIE, a tutor in electronics trouble-shooting (Brown, Burton, and Bell, 1974);
- EXCHECK, a tutor in logic and set theory (Suppes, 1981)

c) geology:

- PROSPECTOR, a consultant in mineral exploration (Duda et al, 1978);

d) computer science: models for database information-retrieval problems, systems for hardware configuration

- LADDER and LIFER, a database management system with a natural language interface (Sacerdoti, 1977);
- R1, system configuration for DEC VAX computers (McDermott, 1981);

e) mechanical engineering:

- SACON, a finite-element analysis system for modeling various structures (Bennett et al, 1978).

This partial list illustrates the many possible applications of expert systems. The common link for all these systems is they implement human expertise that is either in short supply, or is very expensive.

1.1. Expert Systems in Music

In music, expert systems have been developed for compositional and educational purposes. In composition, these systems fall into three categories:

a) systems with rules that are based on an analytical study of a particular composer's style;
b) systems with rules that are based on a contemporary composer's method of composition;
c) systems with rules of harmony and melody to provide options for a composer at each step in the compositional process.

An example of a system in the first category is Ebcioglu's expert system for harmonizing four-part chorales based on Bach's style (1988). Several examples of the second category are described by Hiller (1959 and 1971). These systems are used to compose contemporary electronic music based on rules input by a composer. An example of the third category is Cope's expert system for computer-aided composition (1987).

In education, expert systems have been developed to aid in aural training and musicianship. For example, Gross and Griffin (1982) and Arenson (1984) created ear-training courses with different levels of difficulty based on the

student's past performance in the course. Newcomb (1985) developed an expert tutor in 16th-century counterpoint.

Expert systems in music take advantage of the aspects of music that can be represented using rules. Harmonizing chorales, writing melodies with suitable harmonies, and writing counterpoint are all rule-oriented activities, with rules that have existed for centuries. Educational expert systems in ear-training have more to do with tracking a student's progress than with music per se.

One area of music that has not been implemented in an expert system is musical performance. On the surface, it appears that performance is far from being rule-oriented, because of the required element of human emotion. In reality, however, a large part of performance is rule-oriented, especially if we use one particular expert performer as the basis for the rules. We may not be able to develop rules for emotional expression, but we can codify the specific rules that a particular expert uses in deciding on an interpretation.

1.2. Project Description

1.2.1. General Description

In this research project, we develop and implement a rule-based expert system that determines interpretations of Bach fugues. The rules in the system are based on the expertise of two professional performers.

The input to the system is a numeric representation of the fugue. The processing is based on a language recognizer for rhythmic patterns. Once a particular pattern is recognized in the input, a specific rule is applied, or a procedure is performed. The output is a listing of performance actions. Were a musician to perform these actions while playing the fugue, the performance would sound like one given by our experts.

1.2.2. Tools Used in the Project

The format for the knowledge engineering aspects of this project is based on the work of Orchard and Tausner (1988). They describe three stages of the knowledge engineering process:

a) knowledge elicitation: eliciting the knowledge from the experts;

b) knowledge formalization: organizing the knowledge to reflect the way the expert uses it;

c) knowledge representation: representing the knowledge in appropriate data structures for implementation.

We use this general format in the project. The knowledge formalization stage utilizes the concept of knowledge structures as developed by Klir (1985) and applied to knowledge engineering by Orchard and Tausner. Klir has developed system definitions of knowledge that are invariant across disciplines.

He defines four knowledge structures: source, data, behavior and structure; together, these represent an expert's knowledge. The following definitions are in terms of our expert system. Source contains the vocabulary of the domain, i.e., the words the expert uses to express his/her ideas. Also in source are the defined variables and the possible values for the variables. Data consists of examples that can be measured or recorded. Behavior contains all the rules, or the relationships between the different variables. Structure contains the context or overall picture.

According to Orchard and Tausner, an expert works through a problem in the domain by moving from one knowledge structure to another until a solution is found. We use this formalization in a very general, high-level way, as a basis

for building the expert system. It serves as a paradigm to help organize the knowledge, and it also provides a design for the expert system. More information on knowledge structures is presented later in this thesis.

Another tool used in this project is a program called Music Printer Plus by Temporal Acuity Products (1989). This is used to generate MIDI (Musical Instrument Digital Interface) performances of the fugue before and after the knowledge base is applied. MIDI is a communication standard for digital musical instruments (see Appendix A). Music Printer Plus allows for input of a piece of music by entering it in standard music notation on the screen. A music compiler uses the notation file to generates a MIDI file which can be performed on a synthesizer.

Music Printer Plus also allows input of performance information on the screen, e.g., staccato dots, slurs, ritardandi, accelerandi, etc. Using these symbols, a performer communicates his/her interpretation ideas to other performers. If these performance symbols are included in the notation, the compiler embeds them in the generated MIDI file. When the file is performed on a synthesizer, a much more expressive performance is given.
We use Music Printer Plus in this project to create musical examples. First, we enter the musical notation on the screen.

Music Printer Plus provides a graphic interface where a user can place notes of a particular rhythmic value on a staff. We compile the file generated by this input and listen to the "straight" version using a synthesizer. The next step is to run the programs on the numeric representation of the notation; these programs apply the knowledge. As stated earlier, the output is a listing of performance actions such as "At bar 2, beat 3, play the first 16th staccato and slur the three following 16ths". Using Music Printer Plus performance notation, we add a staccato sign to the first 16th and a slur sign to the three that follow using the graphic interface. Finally, when we re-compile it with this new notation, a much more expressive performance (like the expert's interpretation) is given.

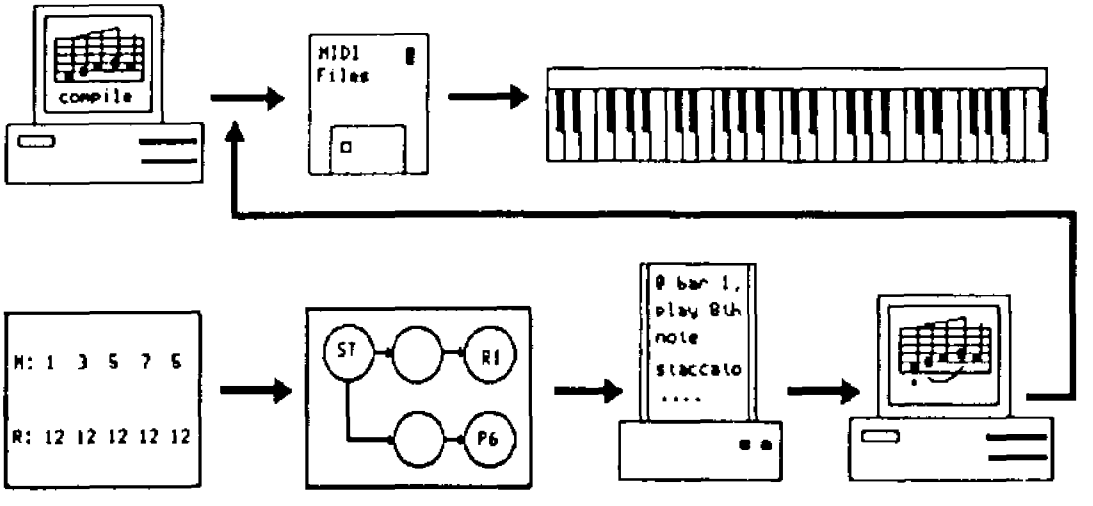


Figure 1. Use of Music Printer Plus

1.2.3. Specific Tasks to be Accomplished

A list of the tasks that are included in this project follows:

1) Decide on a subset of musical pieces and a performance medium.

2) Elicit the musical knowledge from experts.

3) Formalize the knowledge using Klir's knowledge structures.

4) Find a representation for the contents of the knowledge structures and implement the knowledge in data structures.

5) Develop a numeric representation for the music notation which can be used as input to the system.

6) Write a program that simulates the experts' strategy in using the knowledge structures. The output of this program is a series of instructions describing which performance notation symbols available in Music Printer Plus to add to the notation.

7) Represent a piece of music in the numeric format developed in Step 5.

8) Input the notation of the piece from the subset into Music Printer Plus and compile it.

9) Run the program from Step 6 on the numeric representation, add the performance notation symbols to the Music Printer Plus notation as specified in the output of the program, and then compile it again.

10) Play the straight MIDI file and compare it aurally to the enhanced file with the performance information.

11) Do a visual compare of the new edited version of the music notation (with the performance symbols added) against our expert's edition.

12) Do steps 7-11 for other pieces from the subset.

13) Describe future directions, e.g., what needs to be done to further validate the system; what needs to be done to expand the knowledge base to handle pieces outside the subset; what interfaces must be implemented to make the process automatic?

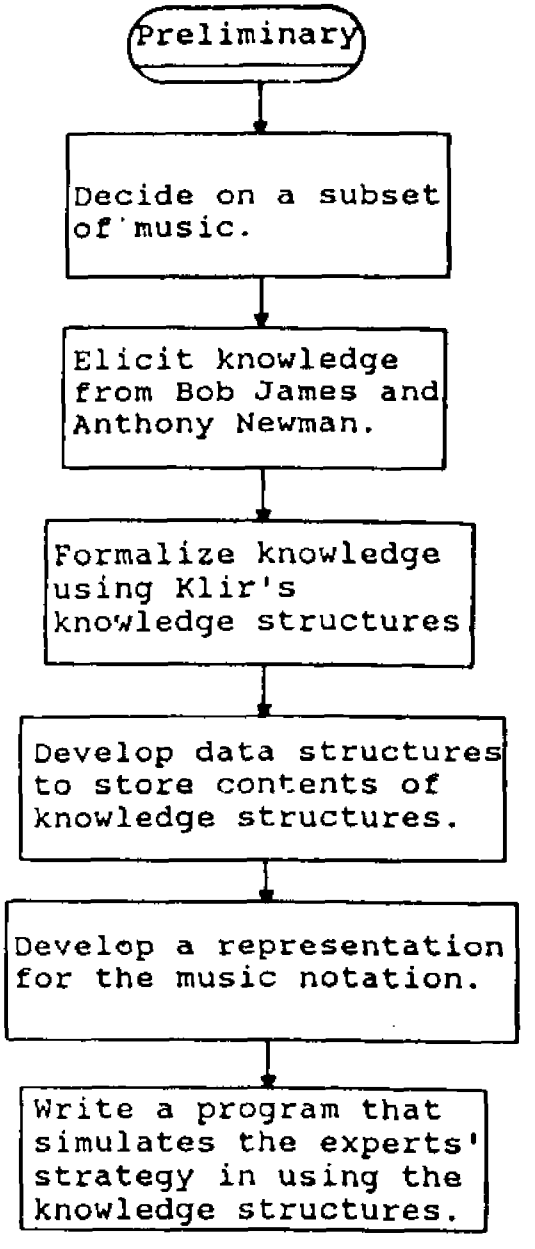


Figure 2. Task Flowchart 1 - Preliminary

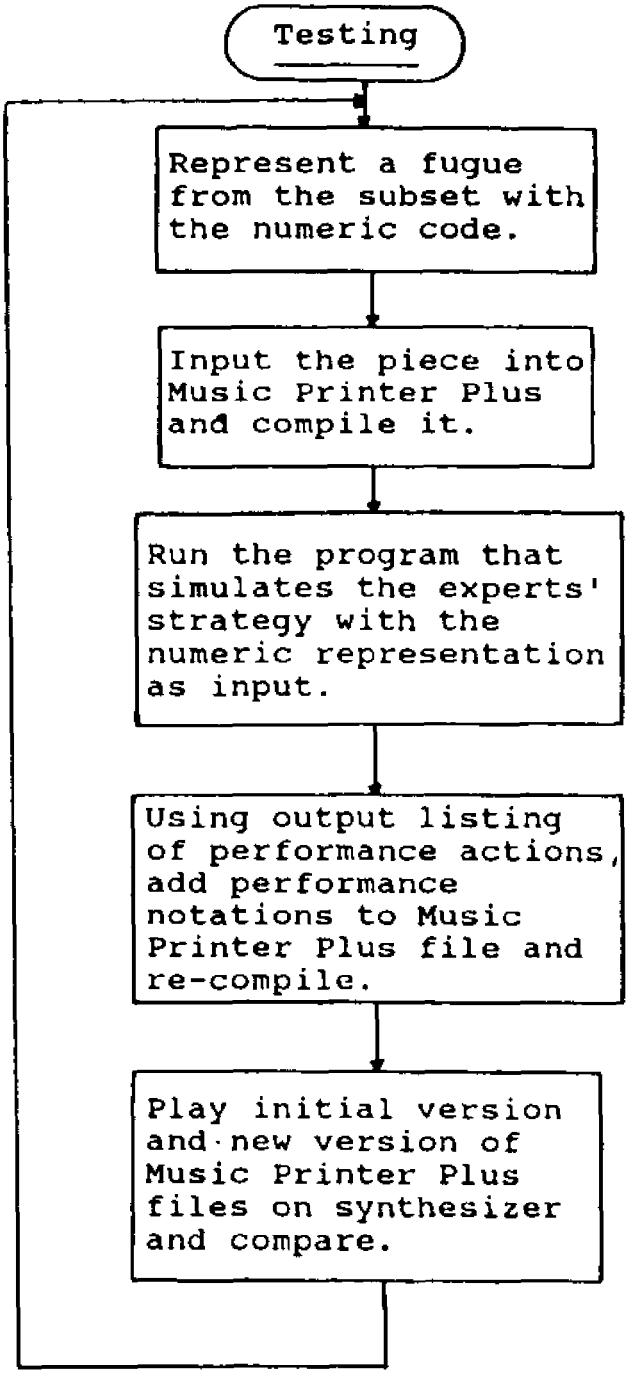


Figure 3. Task Flowchart 2 - Testing

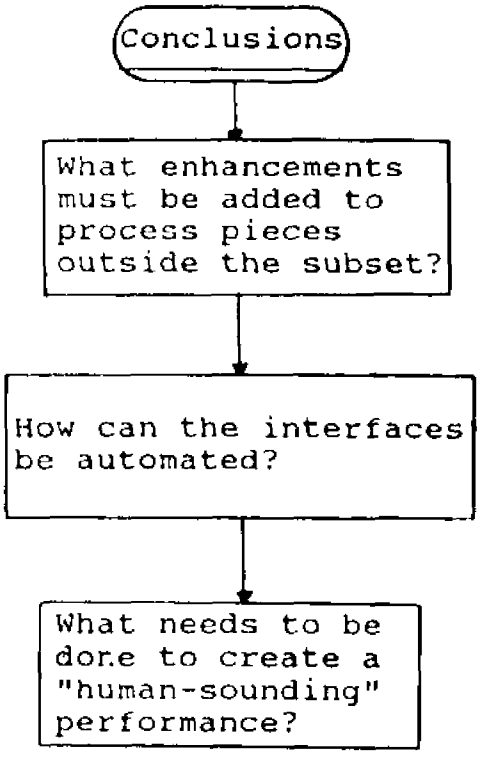


Figure 4. Task Flowchart 3 - Conclusions

The validation of the system for this project consists of a visual compare of the output against a version of the fugue edited by our expert. This constitutes unit testing of the system; aural acceptance testing by the expert is outside the scope of this project.

As can be seen from this list of tasks, this thesis does not automate all the interfaces in this project. The technical tasks involved would be too much of a diversion from the main purpose which is to formalize and represent musical knowledge. Eventually (once all the interfaces are automated), the following scenario would be possible:

1) Input the notation into Music Printer Plus and compile it.

2) Run a translator program that processes the compiled MIDI file, and translates it into the numeric representation.

3) Run the programs that apply the knowledge.

4) Run another translator program that adds the performance information directly to the MIDI file.

Therefore, all one has to do is input the musical notation and the output would be a MIDI file which when performed on a synthesizer, sounds like a particular expert's interpretation. For this project, however, we use human interfaces and Music Printer Plus to facilitate implementation.

1.2.4. Musical Considerations

As for musical considerations in this project, we deal with a subset of the Bach keyboard works as performed on a harpsichord, specifically, the fugues in 4/4 or Common time from the Well-Tempered Clavier Book One:

Fugues:	1	2	3	5	7	8	9	12
	13	16	17	18	20	23	24	

The limitation on meter comes from our experts' belief that meter plays a central role in determining interpretation. This means that the rules elicited for 4/4 fugues differ from those for fugues in 3/4 or 6/8.

Fugues are chosen because they are almost mathematical in the way they evolve. One knows exactly what happens in the piece which makes it much easier to analyze for interpretation than a more free-form piece.

The harpsichord is chosen because this is one of the instruments on which Bach would have performed his fugues. Appendix C contains additional information on musical terms and background.

2. Overview of Related Research

Prior to beginning this project, several resources were investigated to see if any related work has been done in the area of automated performance or expert systems for music. The following overview presents the main categories of research in the musical application of computers.

a) Using computers for cataloging and analyzing musical artifacts.

Musicologists and music theorists have been using computers for this purpose for several years. One of the largest projects is Jan LaRue's Thematic Catalog of 18th-Century Symphonies (1988). LaRue has been a pioneer in the use of computers to catalog themes of musical works in such a way that various queries are possible. For example, it is possible to determine how many 18th-century symphonies are written in each key; or, whether there are any thematic similarities between symphonies written by composers in a particular area. The catalog is also of prime importance for musicologists attempting to establish authorship of anonymous symphonies.

An example in music theory can be found in the work

of Knopoff and Hutchinson (1981). These researchers analyzed the frequency of harmonic dissonance in certain Bach chorales, and discovered a log-normal distribution of the length of inter-dissonance gaps.

Other examples of research in this area are Hill and Ward (1989) who developed relational databases of key text and concepts from musicological articles to aid researchers in finding other relevant work; and, Crerar (1985) who uses analysis of the characteristics of a piece and statistical techniques to determine authorship.

b) Computer-based musical tools for music notations.

Research in this area ranges from software for generating notation (Haus (1983) and Music Printer Plus by Temporal Acuity Products (1989)), to translation programs that generate standard notation from Medieval or Renaissance notations (Crawford and Zeeff (1983), Charnassé and Stépien (1986)).

c) Computer-based musical tools for music education.

Many systems have been implemented for music education, most of which aid in aural training and musicianship. For example, Griffin and Gross (1982)

and Arenson (1984) developed table-driven ear training tutors where the instructor can control all aspects of the training process. Other systems include Newcomb's expert tutor for learning 16th-century counterpoint (1985).

d) Automated composition based on random events or mathematical models.

The first attempts at electronic/computer composition took place in the 1950's. In many studies, the musical decisions were based on random events. This music was not critically acclaimed, but it did lay the groundwork for later approaches. Composers in this area include Stockhausen, Xenakis and Ligeti.

Some examples of more recent work in this area can be found in Lorrain's work in stochastic music (1980), Myhill's work in "controlled indeterminacy" (1979), and Bolognesi's work in automatic composition (1983).

e) Automated composition based on compositional rules.

Also in the 1950's, computer composition based on compositional rules was developed. A style of music

was analyzed either statistically or musically, to extract its characteristics into rules. There has been a lot of work in this area. Refer to Zapirov who uses algorithmic expressions to represent music (1960), and Ebcioglu who developed an expert system for harmonizing four-part chorales based on Bach's style (1988). A good historical survey of automated composition can be found in Hiller (1970).

f) Automated composition based on generative grammars.

The use of generative grammars and formal languages in automated composition has been a very important area of research. For examples, refer to Rader who develops a generative grammar for traditional melodies (1974), Baroni et al, who generate Lutheran chorale melodies in the style of Bach based on a grammar (1975, 1978, 1983), and Holtzman who outlines the use of generative grammars in music (1981).

g) Relationship between music and language.

There has been a great deal of research in the application of formal languages and generative grammars to music. Roads (1988) discusses the

advantages and disadvantages of the use of grammars, and provides an overview of current research.

The starting point of all the research is some kind of analogy to linguistics as presented in Chomsky (1957). The analogies range from direct application (e.g., Bernstein, 1976) to a general application of the concepts (e.g., Lerdahl and Jackendoff, 1977 or Winograd, 1968). A wide range of music can be represented using grammars. Any music that can be segmented and broken down into some kind of hierarchical structure can be described with grammars.

One general problem in the use of grammars in music is the obvious differences between music and natural language. In language, only one sentence is spoken at a time, while in music several melodies may be playing at one time. Language is single-level while music can be multi-level.

The multi-level characteristic of music is further complicated by the problem of context. Since music can be multi-level, any phrase taken out of context can probably be represented using a grammar. However, looking at it in context complicates matters. For example, consider two melodies playing

together. One view of the phrase is the intermingling of the two melodies; another view is the harmony resulting from the weaving of the melodies and the effect of the harmony in moving toward the end of the phrase; still another view is the interaction of the rhythm to push the melodies toward a Cadence; another view is the place of this phrase in the structure of the entire piece.

We may be able to represent the melodies, harmonies and rhythms of a particular piece using grammars; Baroni and Jacoboni (1975, 1978) were successful in developing a set of 56 productions which generated "correct" Lutheran chorale melodies. But the problem arises in trying to represent the context in the grammar. As long as the requirements call for a context-free view of one of the levels or elements of the piece, grammars can be used as a representation. We use grammars in this thesis project, by defining a transition graph that recognizes rhythmic patterns.

h) Synthesis and processing of digital sound.

This category is, by far, the dominant area of research in musical applications of computers. The primary journal in computer music research is the

Computer Music Journal with approximately 80% of its pages dealing with digital sound.

This category of research over the years has laid the foundation for synthesizer hardware, MIDI, and the current revolution in digital instrumentation and recording techniques. It is impractical to give even a general listing of important articles in this category. The early issues of the Computer Music Journal are the main resource for background information. The research on digital sound can be divided into the following sub-categories:

a) Algorithm Design
b) Computer/Synthesizer Architecture
c) Languages for MIDI, Music Programming, and Internal Representations
d) Real-Time Performance
e) Software for Synthesizers, Sequencers and other Music Hardware

i) Recognition and understanding of music, both in hearing and in performance.

Computers and information system concepts have been used to help analyze the musical experience. For example, see McAdams and Bregman who used digital techniques to study how a person hears music (1979).

Also, Rosenbloom (1990) analyzes the brain activity of a performer.

One interesting study in performance has some relevance to this thesis. Sundberg, Askenfelt, and Frydén (1983) studied the differences between a noted piece of music and the music as performed. The authors tried to determine and measure the differences using "pronunciation" rules. These are subjective rules that a performer uses in interpretation. Once the rules were determined, the researchers attempted to measure the quality of performances where the rules were applied in different ways. The idea of rules for interpretation is relevant to this project, but we work only with objective rules, not subjective rules.

Finally, an application of what has been learned about performance was implemented in the Tsukuba Musical Robot (Roads, 1986). This robot was built in Japan and is equipped with a video camera "eye", ten fingers and two feet. It has the general form of a human performer. The robot is capable of performing simple Bach chorale melodies and popular music on a digital organ.

3. Knowledge Elicitation

According to Orchard and Tausner (1988), the first step in the knowledge engineering process is the elicitation of knowledge from experts. Two experts are used in this project: Bob James and Anthony Newman. Bob James is a prominent jazz composer and performer. Anthony Newman is a well-known keyboard performer and musicologist. Bob James is used as the expert on the interpretation decision-making process, while Anthony Newman fills in the details for the particular pieces with which we work.

The expertise was elicited from the following sources:

a) Interviews with each expert.
b) Performance practice manuals (Newman (1985), Barra (1983), Bodky (1960), Donington (1982)).
c) An edition of the Well-Tempered Clavier by Anthony Newman (1983).

The resulting expertise consists of an overall strategy for making interpretation decisions, as well as a series of specific rules on how to play a Bach fugue in the subset.

The elicitation process is quite formal, and at the same time, subjective. The author spent one year in residence learning the art of knowledge elicitation at the CUNY

Graduate Center. A detailed description of the elicitation process is given in Appendix B. In this project, we concentrate on the formalization and representation of musical knowledge, not on the dynamics of elicitation, and the required communication skills.

4. Knowledge Formalization

4.1. General Definitions

As stated in the Introduction, the knowledge structure concept of Klir (1985) is used to formalize the elicited knowledge, and as a basis for the design of the system. We use this concept as a philosophical guide, not in a rigorous, mathematical or systemic sense. The following definitions of Klir's knowledge structures pertain to how an expert reasons about a problem in a particular domain:

a) Source: the vocabulary of special words specific to the domain;
b) Data: the examples of the domain;
c) Behavior: the rules of the domain;
d) Structure: the overall context for the other three knowledge structures. Note that there are many implied models (and meta-system models) that can be referenced. Please refer to Klir (1985) for more information.

As an example, consider a man or woman deciding what to wear to work tomorrow. The contents of the knowledge structures for the problem are as follows:

a) Source: skirts, dresses, suits, shoes, shirts, ties, sweaters, etc.

b) Data: the specific skirts, dresses, suits, shoes, shirts, ties, sweaters that the person owns; the outfits that he/she has worn recently; what outfits people especially like.

c) Behavior: the rules the person uses to make a decision, e.g., all the outfits I wore in the past week are not possible choices; in the summer, I do not wear wool; I don't wear mismatched outfits; I cannot wear jeans - my office does not allow it.

d) Structure: The overall context would be the season and the current weather; the style the person likes, i.e., quiet professional vs. colorful and fashionable. Note that there are many

For this problem, the knowledge structures would contain the information defined above. In order to solve the problem, the person conceptually moves from one knowledge structure to another. One possible strategy for solving the clothes problem is given graphically below, using the "Problem-Solving Roadmap" (Orchard and Tausner, 1988). Notice that we do not include Source as a destination, because the Source vocabulary is really a part of the other three knowledge structures.

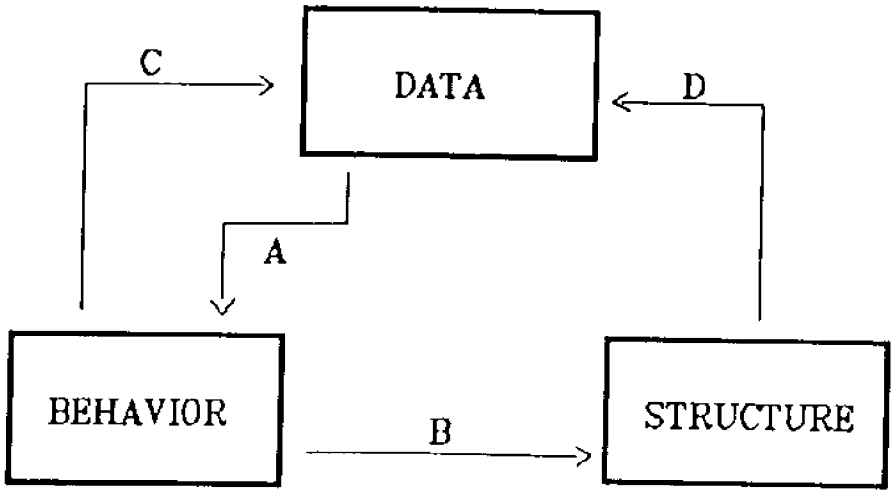


Figure 5. Problem-Solving Roadmap I

The person may begin in Data by staring at the clothes in the closet, and remembering what was worn for the past week. He/she makes a choice, and then goes to Behavior to see if there are any rules that apply to the choice. If the rules all apply favorably, he/she goes to Structure to check the outfit against the context. If the rules in Behavior do not apply, he/she returns to Data to make a different choice. This represents one of many possible ways of solving the problem.

The methodologies used to get from one knowledge structure to another are important because they represent a part of the problem-solving process. The transition from Data to Behavior (a) represents an investigation of the choice made in Data against the rules in Behavior. The transition from Behavior to Structure (b) represents a test of the choice against the context in Structure. The transition from

Behavior to Data (c) occurs because of failure of the initial choice made.

From this example, we see that knowledge structures are a mechanism for representing knowledge about a particular problem. The solution to the problem is found by moving from one knowledge structure to another. There are many possible "Roadmaps" for solving a problem and many possible definitions for the contents of the knowledge structures, the chosen ones depending on the characteristics of the problem and the problem-solver.

The first step in using this formalization to solve a problem is to determine the contents of each knowledge structure, and the transitions between them. This information is gathered in the elicitation process.

4.2. Knowledge Structure Contents for the Expert Performer

As mentioned in the previous section, the first step in the formalization process is to define the contents of the knowledge structures for the problem at hand. Our problem is to determine interpretations of Bach fugues using the expertise of Anthony Newman. Therefore, we define the knowledge structures for this expert.

4.2.1. Source for the Expert Performer

Since Source is the vocabulary of the domain, we must put here all the words (and their meanings) that the performer uses to decide on an interpretation. The following outline defines the basic elements of music, and gives characteristics of each.

Melody: Series of pitches and time values that sound one after the other, and are perceived by the listener as an entity. A melody gives a sense of movement up and down through space as it moves forward in time. This up and down movement is achieved through the use of upward and downward steps or leaps using a major or minor scale. The direction and character of the movement provide expressive meaning. Melody is usually considered a horizontal dimension of music, since the tones sound in succession, in a linear fashion.

The following outline defines the possible characteristics of a melody:

- What is the relationship between successive notes?
  Choices: stepwise, skipping, leaping, chromatic (all half steps); active/stable, articulated/continuous; successive notes part of a repetition or sequence.

- What is the overall melodic pattern?
  Choices: rising, falling, level, wave-like, sawtooth, undulating.

- What is the character of the melody?
  Choices: antecedent-consequent (question-answer), motivic, spinning out, lyrical; vocal quality or instrumental quality; if instrumental, the melody can be made to be idiomatic for a particular instrument.

- Where does the melody come from?
  Choices: composer idea or derived from another piece.

- Where are the highest/lowest points of the melody?
  Choices: Peaks and lows can be anywhere in the piece but are often at climactic points (which usually requires a coming together of different elements at one point in the piece; e.g., a melodic high point may occur on the first beat of a measure, may be held for a long duration, and may be supported by

a meaningful chord in the context of the harmonic language of the piece - all together create the climax).

- What is the tessitura?
  Choices: Make the entire melody high or low or average for the instrument(s)/voice(s) used in the piece.

Harmony: A composite sound made up of two or more tones of different pitch that sound simultaneously. Although a piece of music can consist of a single melody (e.g., Gregorian chant), most Western music depends heavily on harmony to provide structure and expressiveness. Harmony is usually considered a vertical dimension of music since it concerns the tones that sound together in a piece.

A discussion of harmony must include the concept of tonal relationships between different harmonies, and the idea of consonance and dissonance. One of the major characteristics of Western music is its reliance on tonality as an organizational element. Tonal music is characterized by its affirmation of a central tone called a tonic, and the chord built upon that tone. The tonic acts as a center of gravity, a kind of musical home base. It is the point of rest from which the music departs and to which it returns, providing a sense of convincing conclusion.

The main way of establishing a tonal center is through the use of the major-minor scale system. The different tones of a scale have different levels of importance, in terms of their aural relationship to the tonic. For example, the 7th tone (leading tone) has a strong "pull" into the tonic. Likewise, the chord built on the 5th tone (dominant) includes the leading tone, so the dominant chord frequently is the one preceding the tonic. A chord progression that brings the piece to a point of rest is called a cadence; a progression of V - I (dominant to tonic) is quite often used in cadences.

The term tonality refers not only to the relationships between tones and harmonies in one key, but also the relationship between keys. Pieces of music rarely stay in one key for the entire duration; instead, we may shift from one key to another even though the piece begins and ends in the same key. This process of shifting is called modulation. Usually, a modulation in the Baroque period will take us from the tonic to the dominant key, and then back to tonic at the end.

Consonance and dissonance are terms used to describe the "agreeable", stable effect of some harmonies, vs. the "disagreeable", unstable effect of others. This typically depends on whether or not the harmony/tones in question are in the current or closely related key (consonant), or

a distant key (dissonant). Dissonance is important in creating a feeling of tension in the piece, without which the music would be boring and lifeless.

The following outline defines the possible characteristics of harmony in a piece:

- What is the overall harmonic feel of the piece?
  Choices: linear and modal (like Gregorian chant); unified around one key with infrequent, common modulations (to the dominant key for a short period); migrant (frequent modulations to expected places); bifocal (tending to feel as though there are two tonics); expanded (modulating to unexpected places); polycentric or having many tonics; atonal (no tonal feel at all); serial.

- What chords are used?
  Choices: chords within the key; chords within closely related keys (dominant); chords within distantly related keys. Adding dissonances or alterations to different chords.

- How do the harmonies move in relationship to

one another and the other musical elements?

Choices: Different harmonic progressions (ones learned in school vs. more daring ideas); support the melody in an expected or unexpected way; adding chordal sequences and motifs.

- Where are the modulations (if any), and how are they done?

  Choices: Modulatory routes learned in school or daring ideas; using modulation to support melodic and rhythmic ideas.

Rhythm: Like any other sense of time, the passage of musical time is created by change. One tone leading to another in a melody, one chord progressing to another, or moving from one key to another - these and other events work together to create a sense of musical time. The element of rhythm encompasses all aspects of musical time.

The basic unit of musical time is the beat which is just a relative measure of time within the context of a piece of music. Rather than using seconds, we use beats; a note may last a half of a beat, a full beat or several beats. Once the beat of a piece is established, we expect it to continue - it may slow down or speed up, but typically such changes are very gradual since an abrupt change in the

length of a beat would be very unsettling.

Beats are then grouped into units to form a measure. The grouping is based on a meter which indicates how many beats belong in a measure, and which time value gets the beat. A meter of 3/4, for example, indicates that there are three beats to a measure, and a quarter note gets a full beat. Within a measure, certain beats are accented when performed. In 3/4, the first beat is usually accented while in 4/4, the first and third are.

Meters impose regularity and equality on the rhythm of a piece of music. We can slightly alter the rhythmic values of notes within this regular structure to lend tension and interest to the music. The regularity, however, is required in order to recognize the alteration.

Syncopation is a rhythmic technique where the pulse of the meter is deliberately disturbed so that the accented beat of a measure does not occur where the listener expects it. This is another mechanism for providing tension in the music.

The length of the beat determines the tempo, or speed at which the piece should move. If the beat is short, the tempo is fast; if the beat is relatively long, the tempo is slow. Once the tempo is established, it can also be

altered for tension and interest (rubato).

The following outline defines the possible characteristics of rhythm in a piece:

- What is the tempo?

  Choices:

Very Slow -->	Largo (broad)
	Grave (solemn)
Slow ------->	Lento (slow)
	Adagio (leisurely)
Moderate --->	Andante (walking pace)
	Moderato (moderate)
Fast ------->	Allegretto (fast)
	Allegro (cheerful)
Very Fast -->	Vivace (vivacious)
	Presto (very quick)
	Prestissimo (as fast as possible)

- What is the meter? (Note: these are the meters in the Well-Tempered Clavier.)

  Choices: 4/4 3/2 2/4 3/4 6/4 3/8 4/8
  6/8 9/8 12/8 6/16 9/16 12/16

- What is the overall rhythmic feel?

  Choices: homorhythmic (chordal), polyrhythmic (use of different rhythms in different parts,

sometimes strikingly different), polymetric, (simultaneous use of different meters), variant (any combination of the above)

- What rhythmic patterns and durations are used?
  Choices: whole note, half note, quarter note, eighth note, sixteenth note, 32nd note - either alone or in any combination.

- What special rhythmic techniques and notations should be used and where?
  Choices: syncopation, sforzato (forced accent) tempo changes: accelerando (getting faster), ritardando (getting slower), piu lento (slower), piu allegro (faster), fermata (hold indefinitely), rubato.

Sound: Sounds of the same pitch (and octave) differ in sonorous quality depending on the instruments that play them. This is one part of sound/tone color. The other part concerns the texture of the piece itself, the dynamics (degrees of loudness and softness), and the special instrument-specific techniques available to a singer or instrumentalist.

The following outline defines the possible characteristics of sound in a piece:

- What instruments/voices are used?

  Choices: Strings, woodwinds, brass, percussion, voices (soprano, alto, tenor, bass), keyboard - alone or in any combination; some combinations are used for specific styles (e.g. string quartet).

- What is the texture?

  Choices: melody and accompaniment, contrapuntal, homophonic, chordal, imitative polyphony, cantus firmus.

- What are the dynamics of the piece? (Note: this does not really apply to a harpsichord which can play at only one dynamic level.)

  Choices: pianissimo (pp)
  piano (p)
  mezzopiano (mp)
  mezzoforte (mf)
  forte (f)
  fortissimo (ff)

- What are the special effects available on the instrument/voice for which this piece is written?

Choices: depends on the instrument; on a harpsichord, there are different stops or switches that change the sound considerably (lute stop, add more strings, 16 foot stop to add bass).

Also included in Source is the vocabulary for interpretation. We limit this vocabulary to the performance symbols available in Music Printer Plus, since these are the only ones we can use to generate aural output.

- Accents: used to stress or emphasize a certain note or passage to mark its position in the measure or the piece. The following marks indicate degrees of stress:

  > or rf: rinforzando or reinforced

  ^ or sf: sfortato or accented (slightly stronger than >)

  fz: forzando or forced

  sfc: sforzando or forced and emphasized

- Ornaments: trill 𝆗𝆥 , mordent 𝆝 , turn 𝆘 .

- Fermata: a pause or hold 𝄐 𝄒

- Staccato Markings: · detached and distinct
  - sustained but not slurred
  ▾ accented and sustained

- Tenuto: ten. or - held for full time, sustained

- m.m.: ♩ = x Maelzel Metronome: quarter note should equal the time of a click when the metronome is set to x. This is an indication of tempo.

- Accelerando: gradual, though definite increase in speed.

- Ritardando: slower by degrees without a decrease in volume.

Source, therefore, consists of the words the performer uses to communicate his/her interpretation. As in our example above, we do not use Source as a separate knowledge structure, but rather as a part of the other three knowledge structures.

4.2.2. Data for the Expert Performer

Data are examples that can be recorded or measured. In terms of musical performance, Data consists of the music itself as it is performed. A person listening to a piece of music is hearing Data. Therefore, we have Data when the process is complete.

4.2.3. Behavior for the Expert Performer

There are three levels of Behavior for a performer. First, there is the musical score itself as Bach composed it; this is a form of time-invariant Behavior. Second, there is the original music of Bach enhanced with the performance symbols as Anthony Newman would add them giving a new time-invariant Behavior.

Finally, the third level of Behavior consists of rules that guide the performer in deciding on an interpretation. These rules define the expert's behavior while performing. The rules elicited from Anthony Newman concerning fugues from the Well-Tempered Clavier in Common time fall into two categories: how to decide on a tempo, and how to decide on articulations.

Tempo:

Newman (1985) provides a table for determining the tempo of Baroque music based on meter and some other characteristics. The table contains the following fields:

Meter (e.g. 4/4, 6/8, 2/4, etc)
Fastest note value
Where the accents lie
Source tempo
Comments on interpretation

The portion of the table dealing with the fugues of the Well-Tempered Clavier I is given in the table below. (Note: tactus or T ~ 80 on a metronome which corresponds approximately to the human heart beat; a "+" or "-" indicates a slight increase or decrease from 80.)

Sign	Fastest Note Values	Accents	Source Tempo	Comments
C	sixteenth note	1, 3	4 sixteenths = T	1 & 3 held slightly
C	eighth note	1, 3	1 quarter = T+	possible inequality
¢ or 2	eighth note	1	4 eighths = T	
3/4	sixteenth note	1, but irreg.	4 sixteenths = T-⅓ to T-⅓+	
6/4	eighth note	1	dotted half = T	
3/8	sixteenth note	1	6 sixteenths = T	light, Passepied tempo
6/8	sixteenth note	1(3)	6 sixteenths = T	
9/8	sixteenth note	1(3)	6 sixteenths = T	

Figure 6. Tempo Table

Articulations:

According to Newman (1985), the way to decide on articulation and rhythmic alteration is to find the strong/weak elements in the measure, and the strong/weak measures (the concept of strong/weak is discussed in Appendix C under Performance Practice). The rules in Appendix C provide good guidelines for finding strong or weak measures.

The ways to highlight a strong measure or strong beat within a measure are to prolong the downbeat (or another strong beat) slightly, perhaps the value of an additional 32nd note. The time that is added to make this accent is not subtracted elsewhere in the measure. Another way to accentuate a strong measure is to ornament the downbeat chord. Ornaments can also be used to emphasize a strong beat in the measure.

With these general rules in mind, we need to get much more precise. (Note: the following detailed rules pertain to the fugues in Well-Tempered Clavier Book I in Common time.)

The first step is to decide on an articulation for the main melody (or the subject in a fugue).

a) Special Subject Articulation Rules

- The articulations for the subject are played the same way each time the subject appears.

- If certain parts or motives from the subject appear in episodes, they should be played as they were in the subject.
- If the subject opens with a group of 8 16ths (or 32nds) the group of 8 are slurred together.
- A series of more than 2 quarter notes in the subject are played tenuto.
- If the fugue is in a major key and the subject starts with a rest and then a series of 3 8ths, play them all tenuto (if stepwise) and staccato (if there is a leap).
- If the fugue is minor and the subject starts with a rest and then a series of 3 8ths, play them all tenuto.
- Accents fall naturally on 1 and 3, but the note is prolonged only if it is an 8th, not on 16ths.
- In preparation for the prolonged accent on 1 and 3, the previous note, if an eighth note, is played staccato.

The following rules apply after the rules above are tried. If a situation arises where there are two rules (one from above and one from below) both apply, the rule above takes precedence.

The rules that follow are broken down into how to deal with

note values.

b) Quarters

- 2 quarters (only 2) together are slurred, but only if the 2nd quarter is not tied to the next note.
- 2 quarters that are a ½ step apart and the first of the group is altered chromatically, are slurred.

c) Eighths

- If there is a dotted 8th and 16th in beat 2 or 4, trill the 8th; but only if the figure is not accompanying 32nd notes.
- If there is a 16th or dotted 8th and 2 32nd in beat 4 - trill the 16th or 8th; but only if the figure is not accompanying 32nd notes.
- Groups of 4 8th notes are slurred if stepwise; if there is a tie to the first note, or leap of greater than a third between the 1st and 2nd notes, the 1st is played staccato and the remaining 3 are slurred if stepwise; this same rule applies if the first 3 are stepwise and the last is a leap of greater than a third away from the 3rd note: slur first 3 and play last staccato.
- Groups of 2 8ths with a leap of more than a 3rd are both played staccato.

- 2 successive 8ths played together on same pitch are played - · (i.e., tenuto staccato).
- 2 eighths that are a ½ step apart and the first of the group is altered chromatically, are slurred.

d) Sixteenths

- Motivic groupings of 4 16ths are slurred.
- If there is a dotted 8th and 16th in beat 2 or 4, trill the 8th; but only if the figure is not accompanying 32nd notes.
- If there is a 16th or dotted 8th and 2 32nd in beat 4 - trill the 16th or 8th; but only if the figure is not accompanying 32nd notes.
- In groups of 8 16ths, last 7 slurred if there is a leap of more than a third between 1st and 2nd 16th (or a tie between the first and the previous note) and the remaining 16ths are not motivic: the 1st 16th is played staccato if not tied to previous note. The same rule applies to a group of 4 sixteenths.
- All uneven groups of scalar 16ths (e.g., 5 or 6) are slurred.
- A group of 2 sixteenths alone are slurred.

e) 32nds

- All groups of 32nd notes are slurred.

f) Tempo adjustments

- If there is lots of chromaticism in the fugue, tempo is < 80.
- If mostly 8th notes with a few 16ths, tempo > 80.
- If mostly 16th notes, tempo right around 80.
- If subject is mostly quarters, right around 80.
- If lots of 32nd notes, right around 80.

g) Misc.

- A chord of 3 notes or more in one hand on the down beat (beat 1 or 3) is rolled.
- Cadence notes are prolonged slightly.
- Slow down slightly at the second half of the last measure.

4.2.4. Structure for the Expert Performer

Structure for a performer consists of the overall plan of the piece. We are dealing only with fugues in this thesis, so we need to know what a fugue is, and how its structure works. Again, this is one way of interpreting structure as defined in the knowledge structure concept, and applied to music. Other sub-structures are implied, but are not specified.

- Fugue:

A fugue is a polyphonic composition for a fixed number of voices (or melodic parts) built on a single theme called the fugue subject. The subject enters in one voice, is imitated in another voice (the imitation is called the answer), enters in a third, until each voice has stated it. A fugue starts like a round. At frequent intervals throughout the fugue, the subject or answer appears in one voice or another, sometimes in different keys.

The entries of a fugue subject and answer are spaced off by passages called episodes which are made up of melodic fragments, motives, sequences, etc., often derived from the subject or answer. Episodes do not present any really new material, but they provide some relief from the subject.

Another relief is provided by the countersubject(s) which are distinctive melodies which accompany the subject. They tend to stand out from the subject because of their particular melodic or rhythmic profiles. Episodes in fugues sometimes use motives from the countersubject, as well as the subject.

Stretto is a fugal technique where the subject is played

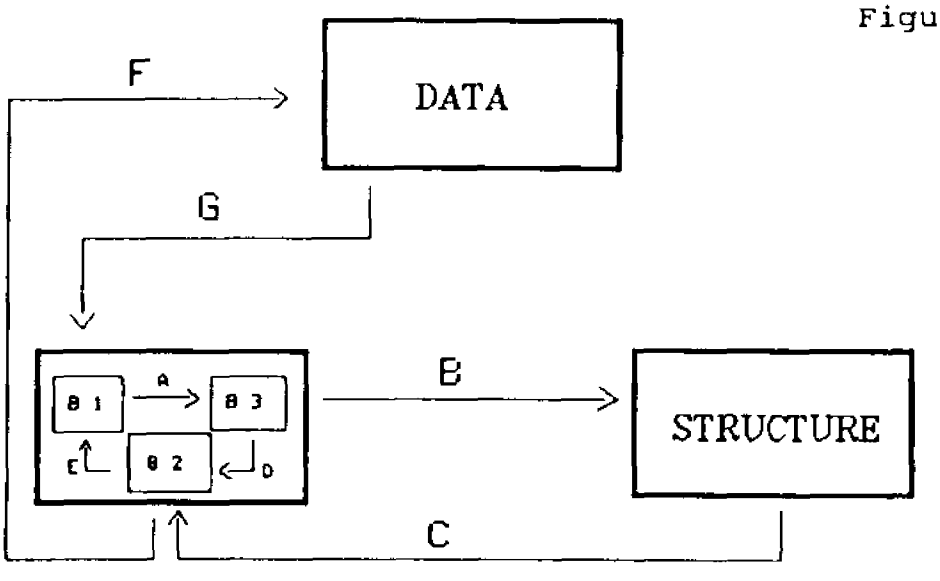
in one voice, and before the voice completes its statement, the subject is started by another voice. A round is really stretto.

See Appendix D for a detailed analysis of one of the fugues from the Well-Tempered Clavier.

4.3. The Expert Performer's Strategy

Now that the knowledge structures have been defined for the problem of musical performance, the strategy for using them and for going from one structure to the other must be analyzed in greater detail. The following strategy is one used by Bob James in the preparation of a piece for performance.

Figure 7.



We start in structure with the analysis of the fugue to determine the subject, answer, countersubject, and the fastest note value. Behavior consists of three levels:

- B1: Bach's musical score
- B2: Edited musical score (i.e., Bach's score enhanced with performance symbols)
- B3: Newman's performance rules.

After completing the analysis in Structure, the expert goes to B1 to look at the meter. Then, he goes to B3 to find the tempo rule that matches the fastest note value and meter. Finally, he goes to B2 to make a note of the tempo. Then, he returns to B1 to look at the pattern of the beginning of the subject. After this, he goes to B3 to see if any rule applies to the rhythmic or melodic pattern; the expert may need additional information in Structure to make the decision. After making the notation in B2, he may test it by going to Data to perform the pattern.

After these preliminary steps, a loop is set up: get the next pattern of notes in B1, look in B3 for rules that apply to the pattern, go to Structure if additional information is needed; go to B2 to make the notation; go to Data to make a performance test if required.

The methodologies associated with each transition are

defined below (please refer to roadmap on page 52):

a: Search

b: Search

c: Edit

d: Edit

e: Get Next Pattern

f: Test

g: Get Next Pattern

We attempt to use the strategy defined above in the overall design of the expert system, with one exception. A performance test is not relevant to us; we produce Data only after the entire editing process is complete.

5. Knowledge Representation

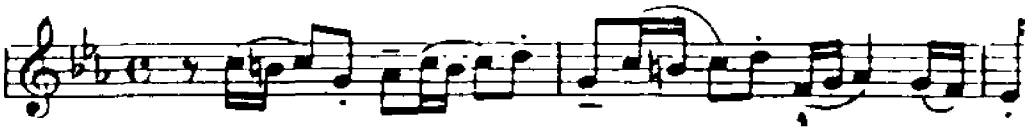
There are three things that must be represented in this project. First, we must find a representation for the music itself. Second, we must represent the information in the knowledge structures. Finally, we must represent the expert's strategy in the program that uses the knowledge structures.

5.1. Representation of the Music Notation

We require a notation that represents the actual music, but in a form that the expert system can use. The representation must provide the following information:

1) note value
2) relative melodic position in the overall melody
3) rhythmic value

The melodic pattern can be represented numerically using half steps (this is the smallest interval available in tonal music). For example, the subject from Fugue #2 from the Well-Tempered Clavier can be represented in this way:



0 -1 0 -5 -3 0 -1 0 2 -5 0 -1 0 2 -7 -5 -3 -5 -7 -8

0 is the starting note, a number indicates the number of half steps away from the starting point, and a positive or negative sign indicates the direction (up or down) from the starting note.

The rhythmic value is specified in terms of 24 "clicks" per quarter note, which is a basic time measure in MIDI. Thus, if a quarter note is 24 clicks, we get the following table which covers all rhythmic values for notes or rests in the Well-Tempered Clavier I fugues in Common Time:

32nd note/rest: 3
dotted 32nd: 4.5
16th note/rest: 6
dotted 16th: 9
8th note/rest: 12
dotted 8th: 18
quarter note/rest: 24
dotted quarter: 36
half note/rest: 48
dotted half: 72
whole note/rest: 96
dotted whole: 144

In musical notation, there are many other permutations possible with ties. A tie from one note value to another is represented by adding the value of the first note to the value of the note tied to it (e.g., a whole note tied to a quarter = 96+24 = 120).

In Appendix E, Fugue #2 in C Minor is represented using this melodic and rhythmic numeric format.

5.2. Representation of the Contents of the Knowledge Structures

Now we turn to the representation of the actual knowledge. Our knowledge currently exists in the four knowledge structures defined in the above sections.

As discussed above, Source is the vocabulary for the performer. We do not use Source as a separate knowledge structure. Its contents are embedded in the other three. Data is the end result of the process, i.e., the aural output generated by Music Printer Plus.

The first level of Behavior (Bach's musical score) is represented using the numeric representation described above, and then stored in a dBASE database. The database for a three voice fugue has the following structure:

Field Name	Length	
Bar	2	
Beat	2	
Note1	4	
RNote1	4	
Note2	4	
RNote2	4	
Note3	4	
RNote3	4	
Melody1	5	(S/A/CS/PT)
Melody2	5	(S/A/CS/PT)
Melody3	5	(S/A/CS/PT)
Step1	4	
Step2	4	
Step3	4	

Figure 10.

The Bar field has the measure number (e.g., from 1 - 31 for Fugue #2); the Beat is 1, 2, 3 or 4 based on which beat in the measure we are on. Note1/2/3 holds the numeric melodic value with the directional sign. Rnote1/2/3 holds the numeric rhythmic value for the note. Melody1/2/3 is either blank, S (subject), A (answer), CS (countersubject), or PT (partial subject or answer statement). Step1/2/3 holds the number of half steps between a note and the next note. The Bar, Beat, Note and Rnote fields are filled in by the user for each fugue in the subset, and represent the notation of the fugues. The Melody and Step fields are filled in by a program which analyzes the fugue and finds the subject/answer/countersubject/partial statements, and calculates the half steps between notes. Refer to Appendix E which illustrates the notation of an entire fugue in this format.

The second level of Behavior is the output listing which contains the editing instructions for enhancing the sheet music. Refer to Appendix F which contains an output listing for Fugue #2 in C Minor. Appendix D contains Newman's edited version of this piece.

The third level of Behavior (the rules) are represented in a dBASE database. The listing below gives all the rules that are a part of Behavior. They are numbered for lookup purposes.

RULE1 IF Subject opens with group of 8 16ths or 32nds, THEN Slur the group of 8 notes together.

RULE2 IF Series of more than 2 quarter notes are in the subject, THEN Play the quarter notes tenuto.

RULE3 IF Fugue is in a major key, begins with a rest then 3 stepwise 8ths, THEN Play the 8ths tenuto.

RULE4 IF Fugue is in a major key, begins with a rest then 3 leaping 8ths, THEN Play the 8ths staccato.

RULE5 IF Fugue is in a minor key, begins with a rest then a series of 3 8ths, THEN Play the 8ths tenuto.

RULE6 IF Beat is 1 or 3 and current note is greater than a 16th, THEN Prolong the downbeat note slightly.

RULE7 IF Next beat is 1 or 3 and next note is greater than 16th and current note is 8th, THEN Play the last 8th of the beat staccato.

RULE8 IF 2 (& only 2) quarters are together in the subject & 2nd is not tied to next note, THEN Slur the 2 quarters.

RULE9 IF 2 quarters in the subject are 1/2 step apart and first is altered chromatically, THEN Slur the 2 quarters.

RULE10 IF A dotted 8th & 16th are in beat 2 or 4 and accompanying figure is not a 32nd, THEN Trill the 8th note.

RULE11 IF A 16th or dotted 8th & 2 32nds are on beat 4 & accompanying figure is not 32nds, THEN Trill the dotted 8th note.

RULE12 IF There are 4 stepwise 16th notes, THEN Slur the 4 16ths.

RULE13 IF There are 4 16ths and a tie to the first, THEN Slur the 16ths after the long note.

RULE14 IF There are 4 16ths and a leap of greater than a third between the 1st & 2nd, THEN Play 1st 16th staccato and slur last 3 16ths.

RULE15 IF There are 4 16ths & the first 3 are stepwise & the last a leap more than a 3rd, THEN Slur the 1st 3 16ths and play the last staccato.

RULE16 IF There are 2 8ths with a leap of more than a 3rd between them, THEN Play both 8ths staccato.

RULE17 IF There are 2 8ths played together on the same pitch, THEN Play both 8th staccato tenuto.

RULE18 IF There are 2 8ths that are 1/2 step apart & the first is altered chromatically, THEN Slur the 2 8ths.

RULE19 IF There is a group of 8 16ths and a leap of more than a 3rd between 1st & 2nd 16th, THEN Play 1st 16th staccato and slur last 7 16ths.

RULE20 IF There is a group of 8 16ths and a tie to the first 16th, THEN Slur the 16ths after the long note.

RULE21 IF There is a group of 3, 5, 6 or 7 stepwise 16ths, THEN Slur the 16ths together.

RULE22 IF There is a group of 2 16ths alone, THEN Slur the 2 16ths together.

RULE23 IF There is a group of 2 or more 32nds alone, THEN Slur the 32nds together.

RULE24 IF There are a group of 3 or 7 16th notes following a rest, THEN Slur the 16ths following the rest.

RULE25 IF The final note of the subject or answer is on beat 1 of a measure, THEN Include this last note in the next sequence.

RULE26 IF There is any group of 16ths following a tied note, THEN Slur the group of 16ths following the long note.

The Structure of the fugue is represented in the Melody fields in the database holding the Data. An analysis program places the letters "S" for subject, "A" for answer, "CS" for counter-subject, or "PT" for partial subject statement in the MELODY fields. Refer to Appendix E.

5.3. Representation of the Expert's Strategy

5.3.1. Overview

The basic premise of our implementation is to simulate the expert's strategy in making interpretation decisions. We have defined the knowledge structures and found methods for representing the knowledge in them. Now, we write a program that uses the knowledge structures and simulates the expert's transitions between them.

As discussed above, the expert begins by analyzing the piece and determining its structure. Then, he sets the tempo based on the characteristics of the notation. We do the same thing by analyzing the numeric notation to determine the primary melodies, and then we run a program that searches the tempo table. The output of the analysis process (S/A/CS/PT) is placed in the database Melody fields.

The expert's next step is to go to B1 to begin making interpretation decisions. He looks at a part of the opening phrase. We begin by opening the fugue database containing the numeric representation and the results of the analysis program. Then, we take the first note and begin processing it.

The expert goes to B3 and Structure to determine if any rules apply to the notes being processed. If a rule does apply, output is generated in B2. Then, he returns to B1 for more notes, and continues. Our implementation of this is through the use of a transition graph. The rhythmic pattern is used to traverse the graph. If we end on a node with a RULE number, we do a lookup in the RULE database and print out the action. If we end on a node with a PROC number, we do the procedure which simulates additional trips to B2 and Structure. Finally, if we end on a node with neither, the search has failed, so we continue with the next set of patterns in B1.

On the technical side, since the knowledge structures are stored in dBASE databases, we use the dBASE programming language as implemented in the Clipper compiler. We now discuss these programs in detail.

5.3.2. Analysis Program

The first step of the analysis process is to determine the pattern of half steps between each successive note. These half step patterns are used to locate the subject and answer statements throughout the fugue. This program fills in the STEP1, STEP2, and STEP3 fields of the database by using the values in NOTE1, NOTE2 and NOTE3. All the program does is subtract the NOTE value in the next record (the

next note) from the current one. The flowchart below illustrates the process. This process in implemented in STEPS.PRG. (Refer to Appendix G for source code listings.)

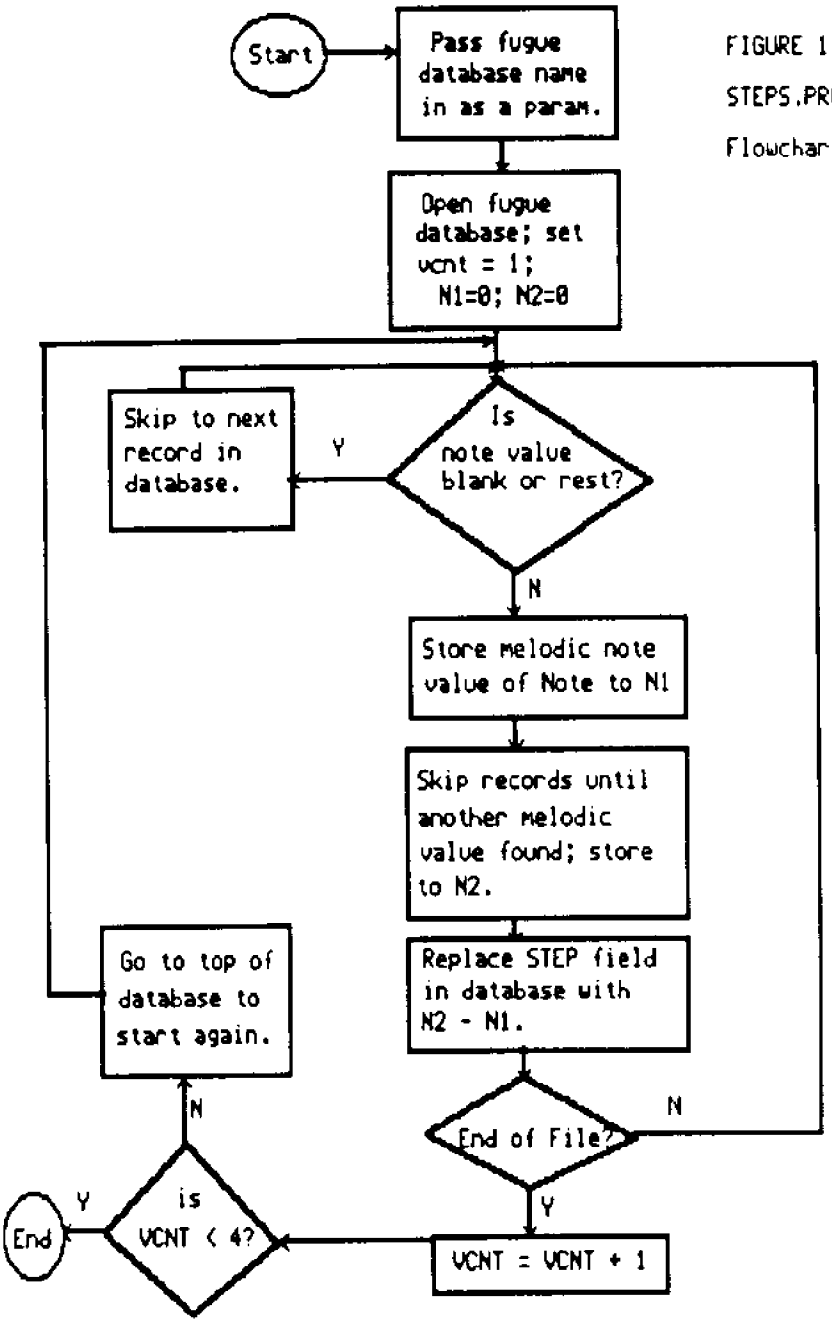


FIGURE 11

STEPS.PRG

Flowchart

Control then passes from the STEPS program to the ANALYZE program which works with the numeric representation stored in the database. The analyze program must find the subject, answer, countersubject (if there is one), and any partial statements of the subject or answer. It also determines if the fugue is in a major or minor key, and finds a base tempo for the piece.

The general design is as follows: To find whether the fugue is in a major or minor key, we count the number of occurrences of minor thirds (NOTE value of 3) versus major thirds (NOTE value of 4). If there are more minor thirds, the fugue is in a minor key; if there are more major thirds, the fugue is in a major key. To determine a base tempo, we need the meter (which is always C or 4/4 in the test subset), and the fastest note value. The fastest note value can be obtained by looking at the RNOTE fields for the smallest value. Then, we do a lookup in the TEMPO file for the meter and fastest note value, and output the instruction for that combination.

To find the occurrences of the different melodies, the following process is performed: we know that the subject is the first melody to sound. We determine the end of it by finding the point where a second voice begins to play. This second voice marks the end of the subject, and the beginning of the answer. The melody accompanying the answer may be a

countersubject. Using these general rules, we can find the patterns making up these melodies, and mark them (S/A/CS) in the fugue database.

We also can find partial statements of subjects or answers. These are inexact statements, but close enough to sound like the melody. We find partials by looking at the rhythmic values of a string of notes and the direction signs, rather than looking at the exact melodic pattern. If a melody has the same rhythm and goes up and down in the same way as the subject, then it sounds like the subject, even though the exact notes may be different.

The following flowchart illustrates the design of the ANALYZE program. The source code is listed in Appendix G.

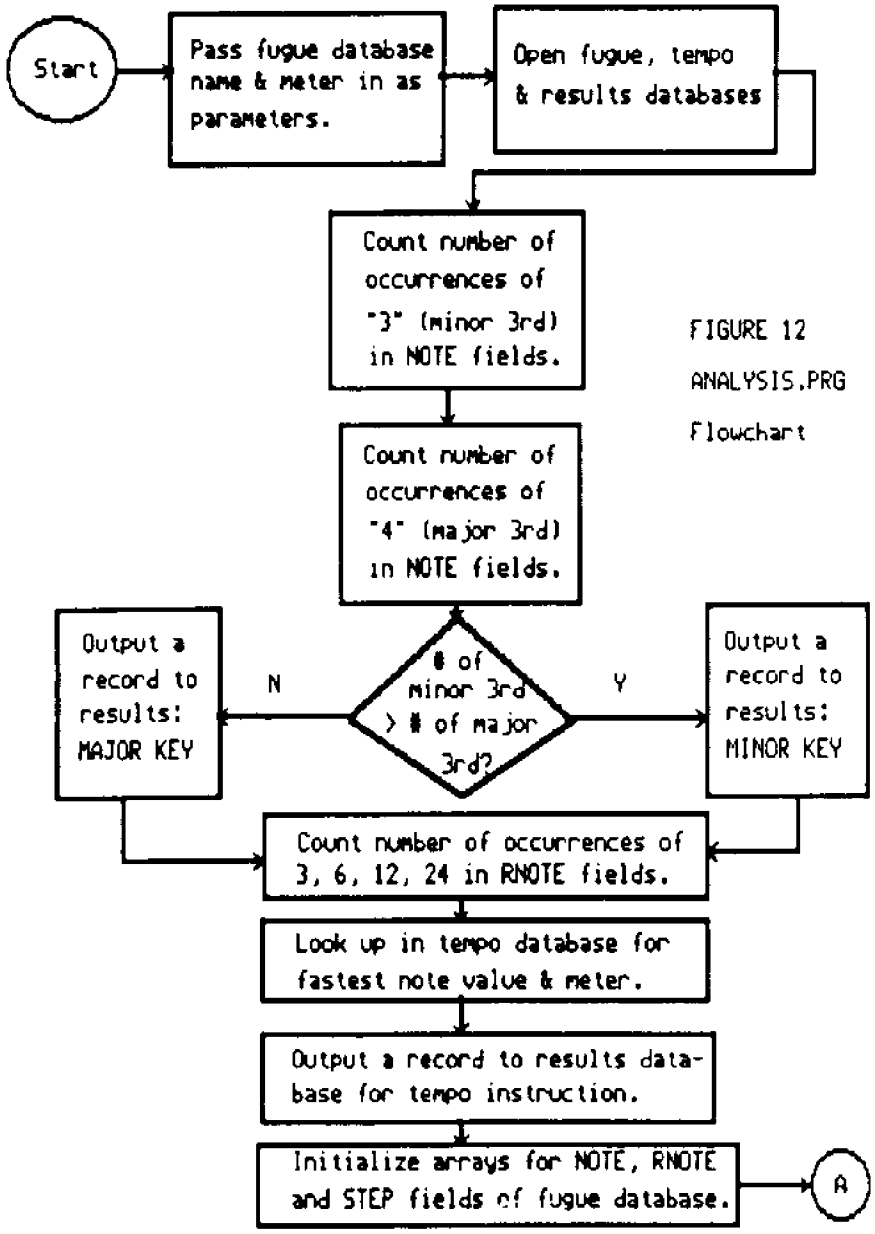
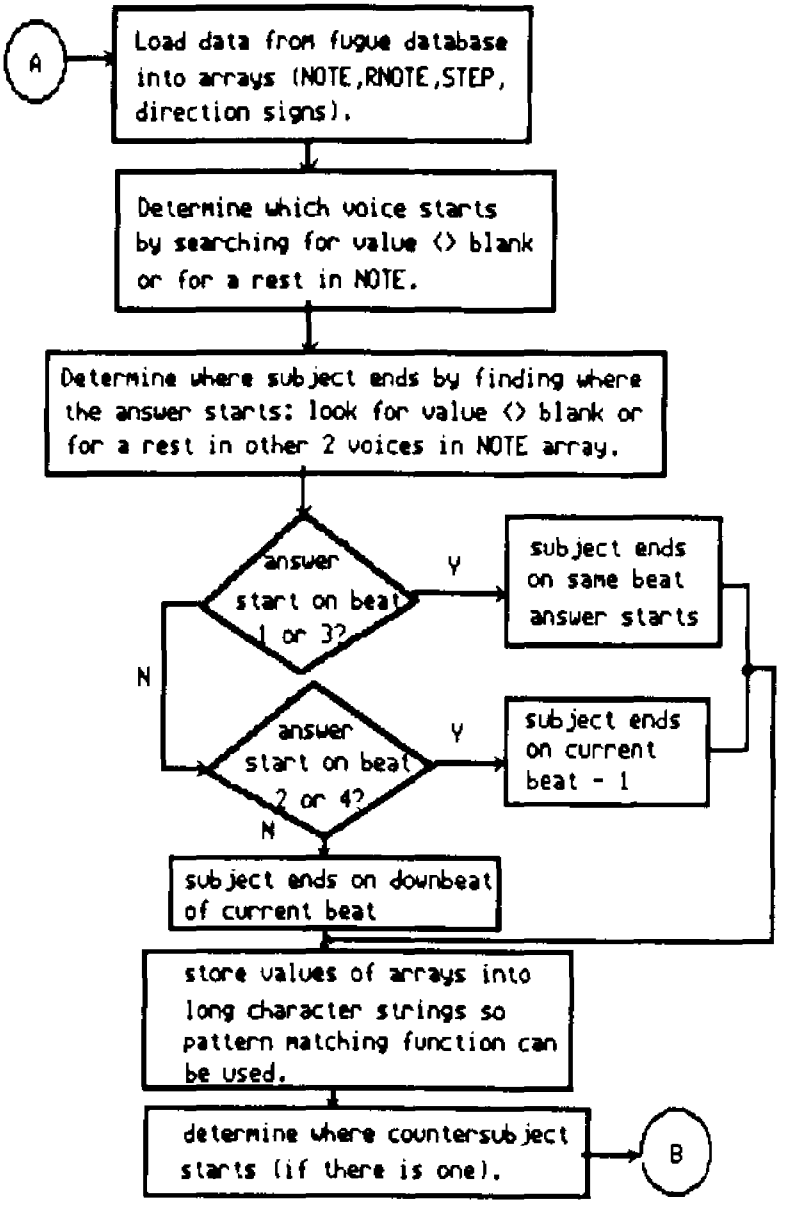
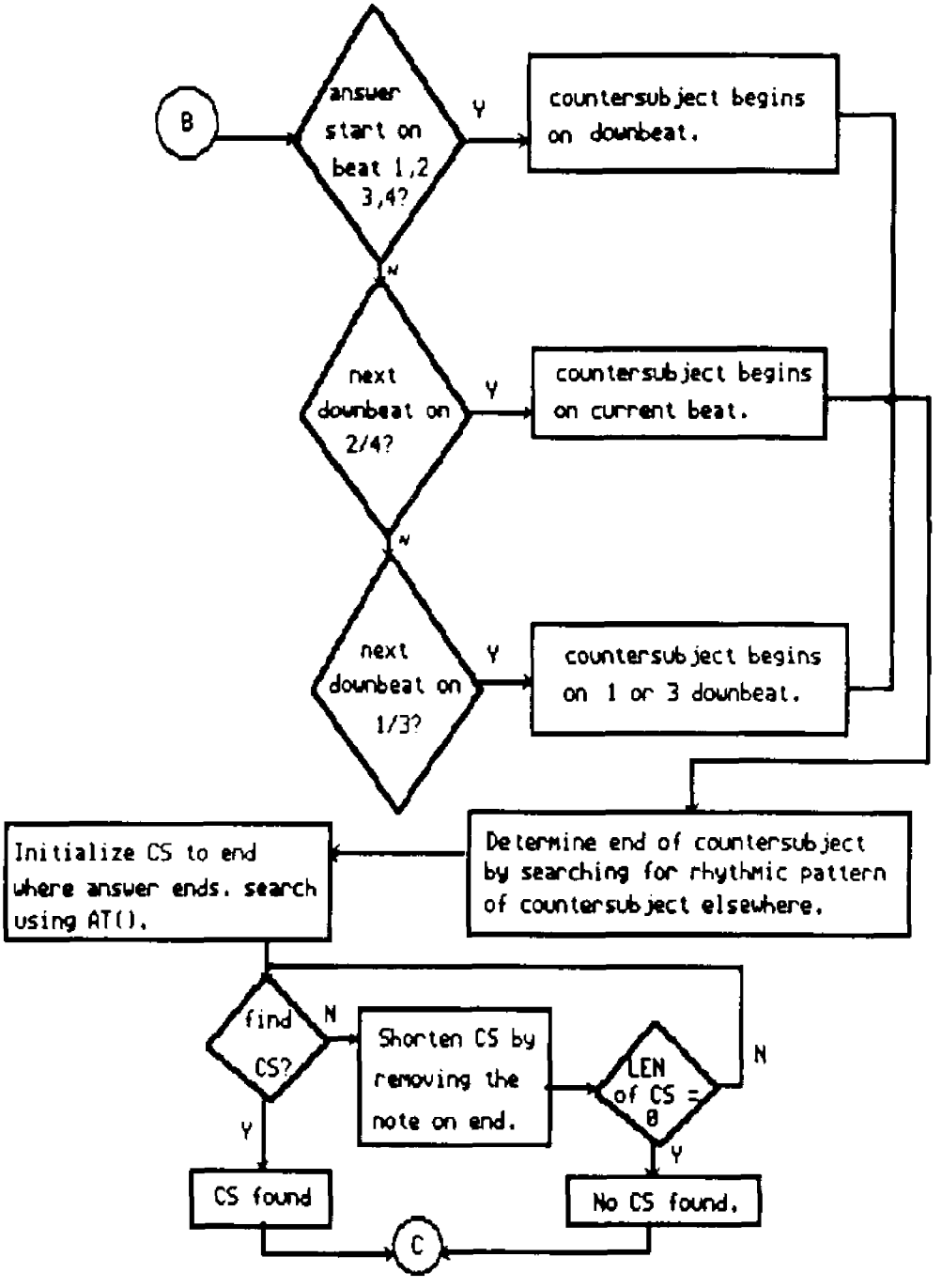


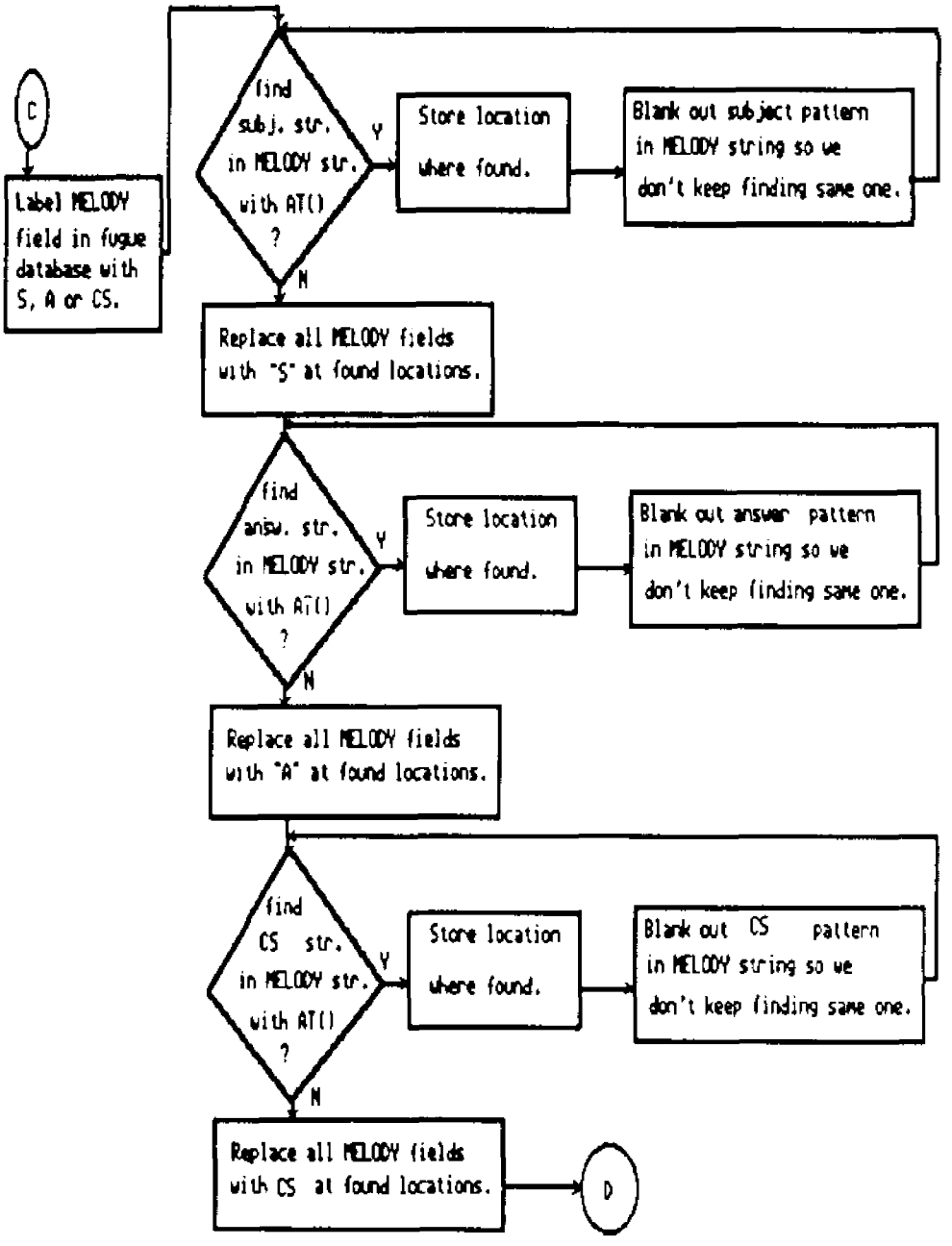
FIGURE 12

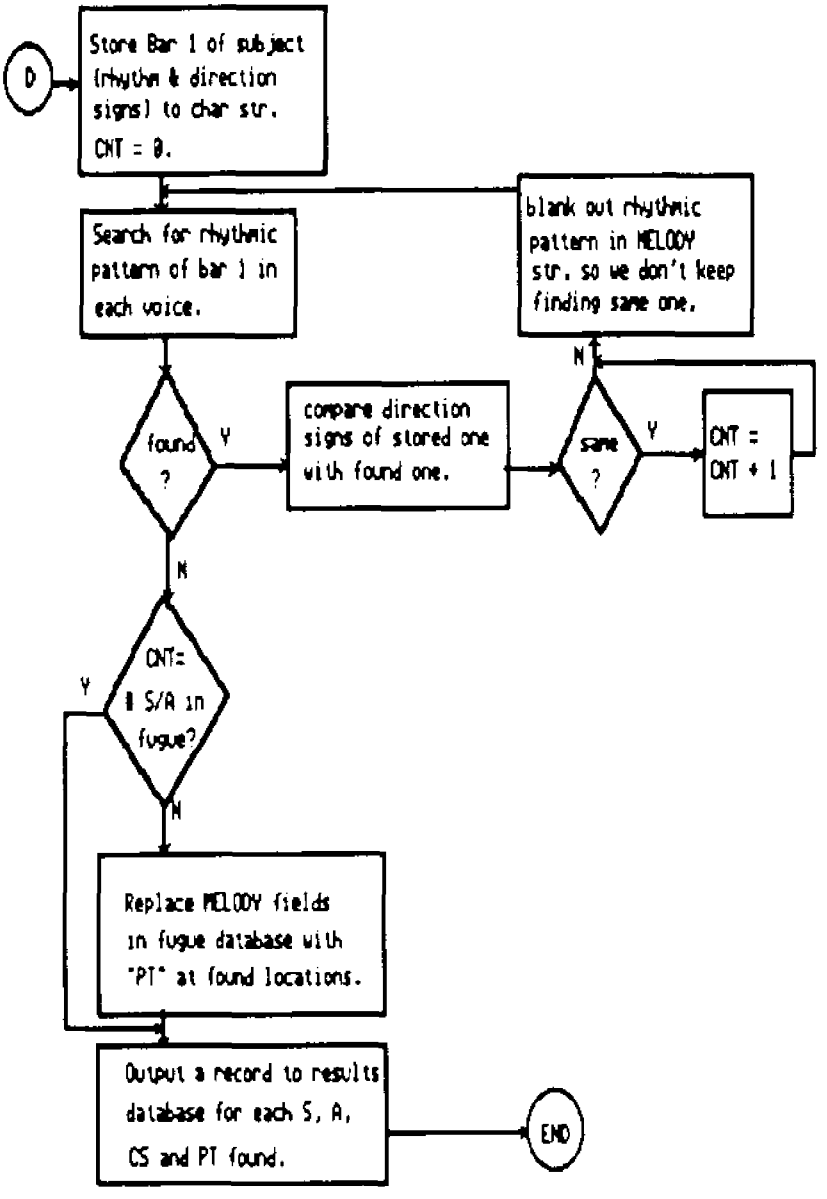
ANALYSIS.PRG

Flowchart


A
Load data from fugue database into arrays (NOTE,RNOTE,STEP, direction signs).
Determine which voice starts by searching for value <> blank or for a rest in NOTE.
Determine where subject ends by finding where the answer starts: look for value <> blank or for a rest in other 2 voices in NOTE array.
answer start on beat 1 or 3?
Y
subject ends on same beat answer starts
N
answer start on beat 2 or 4?
Y
subject ends on current beat - 1
N
subject ends on downbeat of current beat
store values of arrays into long character strings so pattern matching function can be used.
determine where countersubject starts (if there is one).
B


B
answer start on beat 1,2 3,4?
Y
countersubject begins on downbeat.
N
next downbeat on 2/4?
Y
countersubject begins on current beat.
N
next downbeat on 1/3?
Y
countersubject begins on 1 or 3 downbeat.
Determine end of countersubject by searching for rhythmic pattern of countersubject elsewhere.
Initialize CS to end where answer ends. search using AT().
find CS?
N
Shorten CS by removing the note on end.
LEN of CS = 0
N
Y
Y
CS found
No CS found.
C


C
Label MELODY field in fugue database with S, A or CS.
find subj. str. in MELODY str. with AT() ?
Y
Store location where found.
Blank out subject pattern in MELODY string so we don't keep finding same one.
N
Replace all MELODY fields with "S" at found locations.
find answ. str. in MELODY str. with AT() ?
Y
Store location where found.
Blank out answer pattern in MELODY string so we don't keep finding same one.
N
Replace all MELODY fields with "A" at found locations.
find CS str. in MELODY str. with AT() ?
Y
Store location where found.
Blank out CS pattern in MELODY string so we don't keep finding same one.
N
Replace all MELODY fields with CS at found locations.
D


D
Store Bar 1 of subject (rhythm & direction signs) to char str. CNT = 0.
Search for rhythmic pattern of bar 1 in each voice.
found ?
Y
compare direction signs of stored one with found one.
same ?
Y
CNT = CNT + 1
N
blank out rhythmic pattern in MELODY str. so we don't keep finding same one.
N
CNT= # S/A in fugue?
Y
N
Replace MELODY fields in fugue database with "PT" at found locations.
Output a record to results database for each S, A, CS and PT found.
END

5.3.3. Transition Graph Implementation

The transition graph recognizes rhythmic patterns. Once a pattern is recognized, an action is performed. This action is stored in the node following the final note of the pattern. The transition graph is illustrated below. The values on the transitions correspond to rhythmic values as stored in the RNOTE fields of the fugue database. The actions in the nodes are either "R" (RULE) or "P" (PROC). A listing of the RULE database is also below.

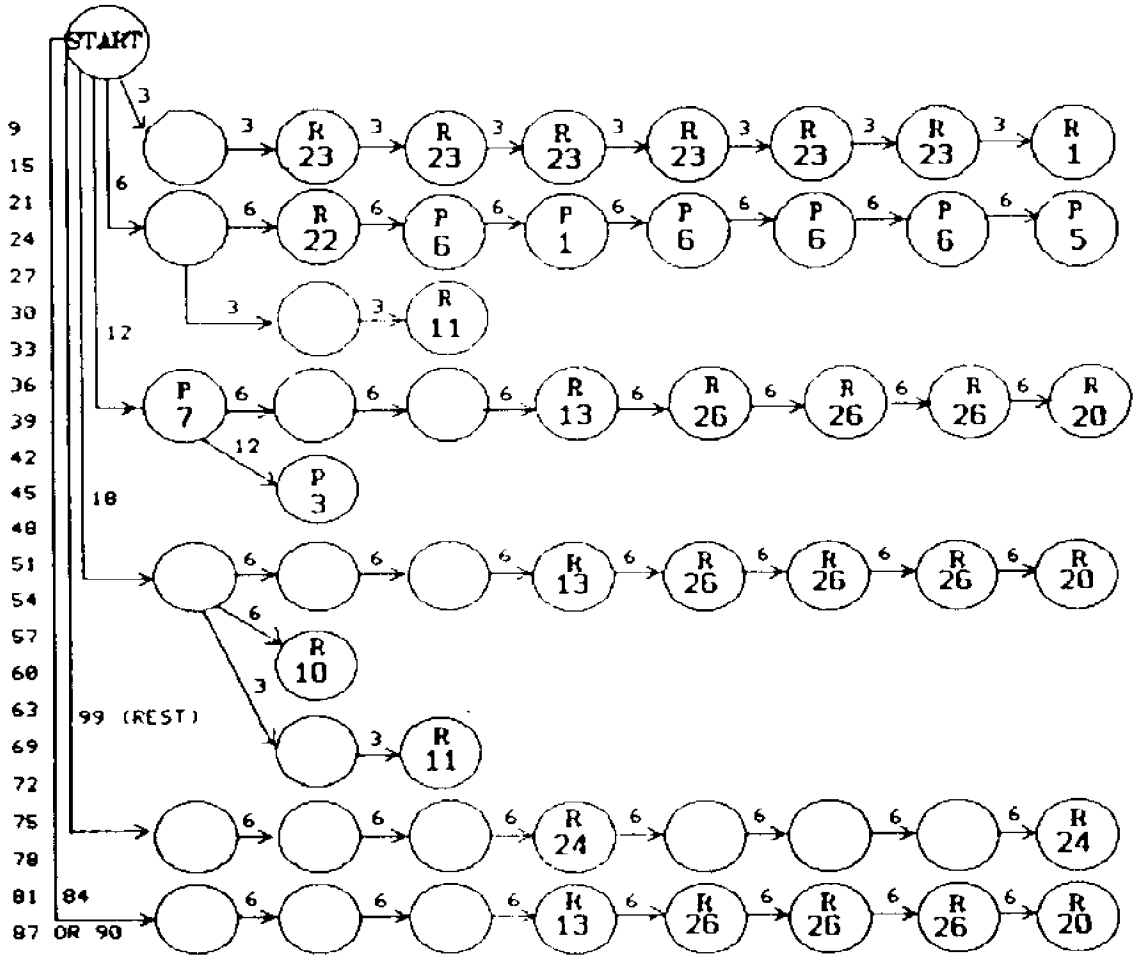


Figure 13. Transition Graph

A representation of this graph for use by the program that traverses it, is stored in a dBASE file:

	NODE1	NODE2	NODE3	NODE4	NODE5	NODE6	NODE7	NODE8
1:	3	3	3	3	3	3	3	3
2:	6	6	6	6	6	6	6	6
3:	6	3	3					
4:	9	6	6	6	6	6	6	6
5:	12	12						
6:	12	6	6	6	6	6	6	6
7:	15	6	6	6	6	6	6	6
8:	18	6	6	6	6	6	6	6
9:	18	6						
10:	18	3	3					
11:	21	6	6	6	6	6	6	6
12:	24	6	6	6	6	6	6	6
13:	27	6	6	6	6	6	6	6
14:	30	6	6	6	6	6	6	6
15:	33	6	6	6	6	6	6	6
16:	36	6	6	6	6	6	6	6
17:	39	6	6	6	6	6	6	6
18:	42	6	6	6	6	6	6	6
19:	45	6	6	6	6	6	6	6
20:	48	6	6	6	6	6	6	6
21:	51	6	6	6	6	6	6	6
22:	54	6	6	6	6	6	6	6
23:	57	6	6	6	6	6	6	6
24:	60	6	6	6	6	6	6	6
25:	63	6	6	6	6	6	6	6
26:	66	6	6	6	6	6	6	6
27:	69	6	6	6	6	6	6	6
28:	72	6	6	6	6	6	6	6
29:	99	6	6	6	6	6	6	6

Figure 14. Database for Transition Graph

	ACT1	ACT2	ACT3	ACT4	ACT5	ACT6	ACT7	ACT8
1:		RULE23	RULE23	RULE23	RULE23	RULE23	RULE23	RULE1
2:		RULE22	PR6	PR1	PR6	PR6	PR6	PR5
3:			RULE11					
4:				RULE13	RULE26	RULE26	RULE26	RULE20
5:	PR7	PR3						
6:				RULE13	RULE26	RULE26	RULE26	RULE20
7:				RULE13	RULE26	RULE26	RULE26	RULE20
8:				RULE13	RULE26	RULE26	RULE26	RULE20
9:		RULE10						
10:			RULE11					
11:				RULE13	RULE26	RULE26	RULE26	RULE20
12:				RULE13	RULE26	RULE26	RULE26	RULE20
13:				RULE13	RULE26	RULE26	RULE26	RULE20
14:				RULE13	RULE26	RULE26	RULE26	RULE20
15:				RULE13	RULE26	RULE26	RULE26	RULE20
16:				RULE13	RULE26	RULE26	RULE26	RULE20
17:				RULE13	RULE26	RULE26	RULE26	RULE20
18:				RULE13	RULE26	RULE26	RULE26	RULE20
19:				RULE13	RULE26	RULE26	RULE26	RULE20
20:				RULE13	RULE26	RULE26	RULE26	RULE20
21:				RULE13	RULE26	RULE26	RULE26	RULE20
22:				RULE13	RULE26	RULE26	RULE26	RULE20
23:				RULE13	RULE26	RULE26	RULE26	RULE20
24:				RULE13	RULE26	RULE26	RULE26	RULE20
25:				RULE13	RULE26	RULE26	RULE26	RULE20
26:				RULE13	RULE26	RULE26	RULE26	RULE20
27:				RULE13	RULE26	RULE26	RULE26	RULE20
28:				RULE13	RULE26	RULE26	RULE26	RULE20
29:				RULE24				RULE24

The program that traverses this graph begins by finding the first rhythmic value of the pattern in the database. For example, consider the rhythmic pattern from the fugue database: 9 6 6 6 24 (dotted 16th, 16th, 16th, 16th, quarter). The lookup on the NODE1 value of 9 in the graph database puts us on record 4. Then, we loop through the subsequent values in the fugue database and compare them to the record that we are on in the graph database. Everything is fine until we hit 24. This is not in the graph pattern. Therefore, we look at the ACTION node on record four for the

last note to match (which is ACTION4). This tells us to apply RULE13. We look up RULE13: IF There are 4 16ths and a tie to the first, THEN Slur the 16ths after the long note. The output is the THEN part of the rule. If the ACTION had been a PROC, then the appropriate procedure would have been run. Procedures simulate additional transitions to Behavior and Structure. They are required in situations where more than one rule applies and additional data must be analyzed to decide on the appropriate rule. The list below indicates which rules are processed in each procedure. Please refer to the rule listing above.

PROC1: Decides between RULE12, RULE14, RULE15 depending on whether a series of 16th notes are stepwise, or if there is a leap between the first and second, or third and fourth.

PROC2: Decides between RULE3, RULE4, RULE5 depending on the key of the fugue and if a rest-8th-8th-8th series is stepwise or not.

PROC3: Decides between RULE7, RULE16, RULE17, RULE18 depending on whether a series of 2 8ths is stepwise, leaping or on the same pitch.

PROC4: Decides between RULE8, RULE9 depending on the relationship between two quarters.

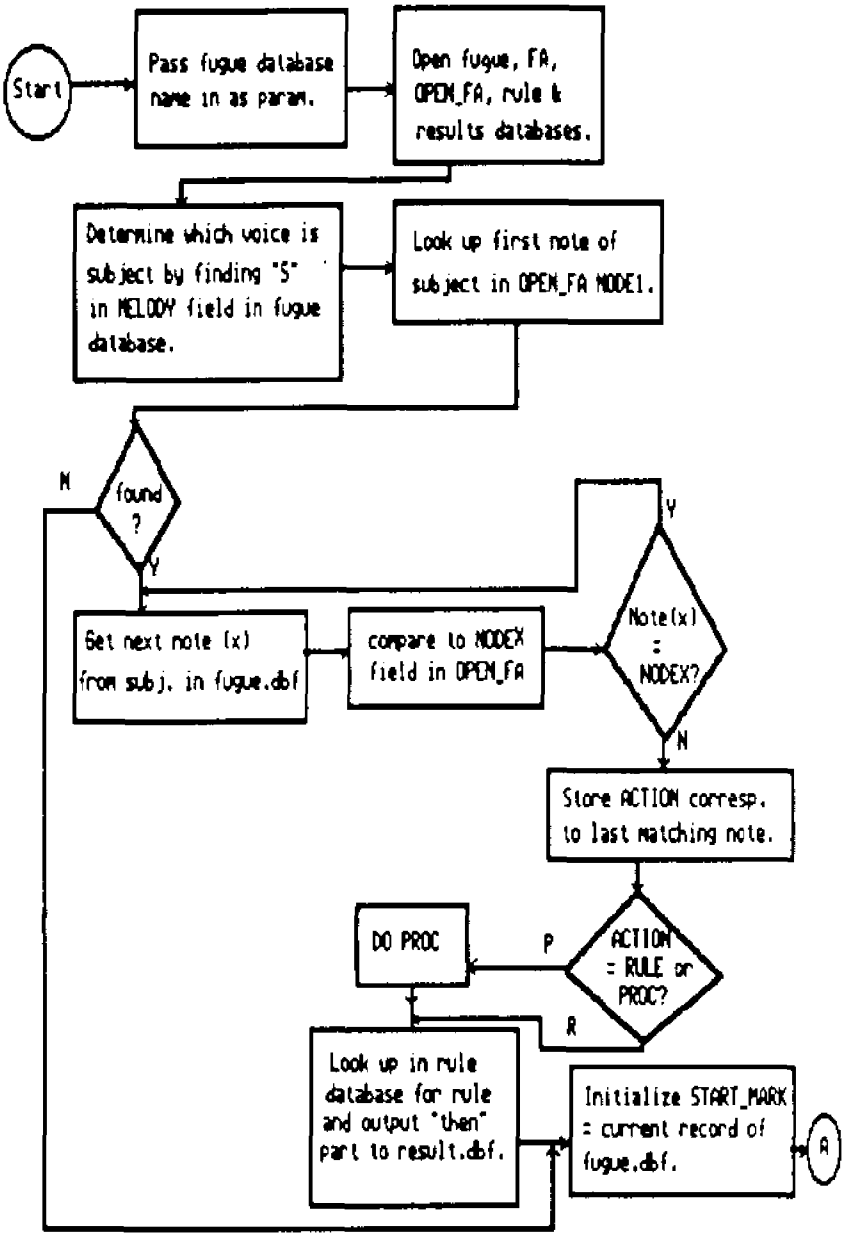
PROC5: Decides between RULE1, RULE19, RULE24, RULE25 depending on the step/leap sequence of a series of 16ths.

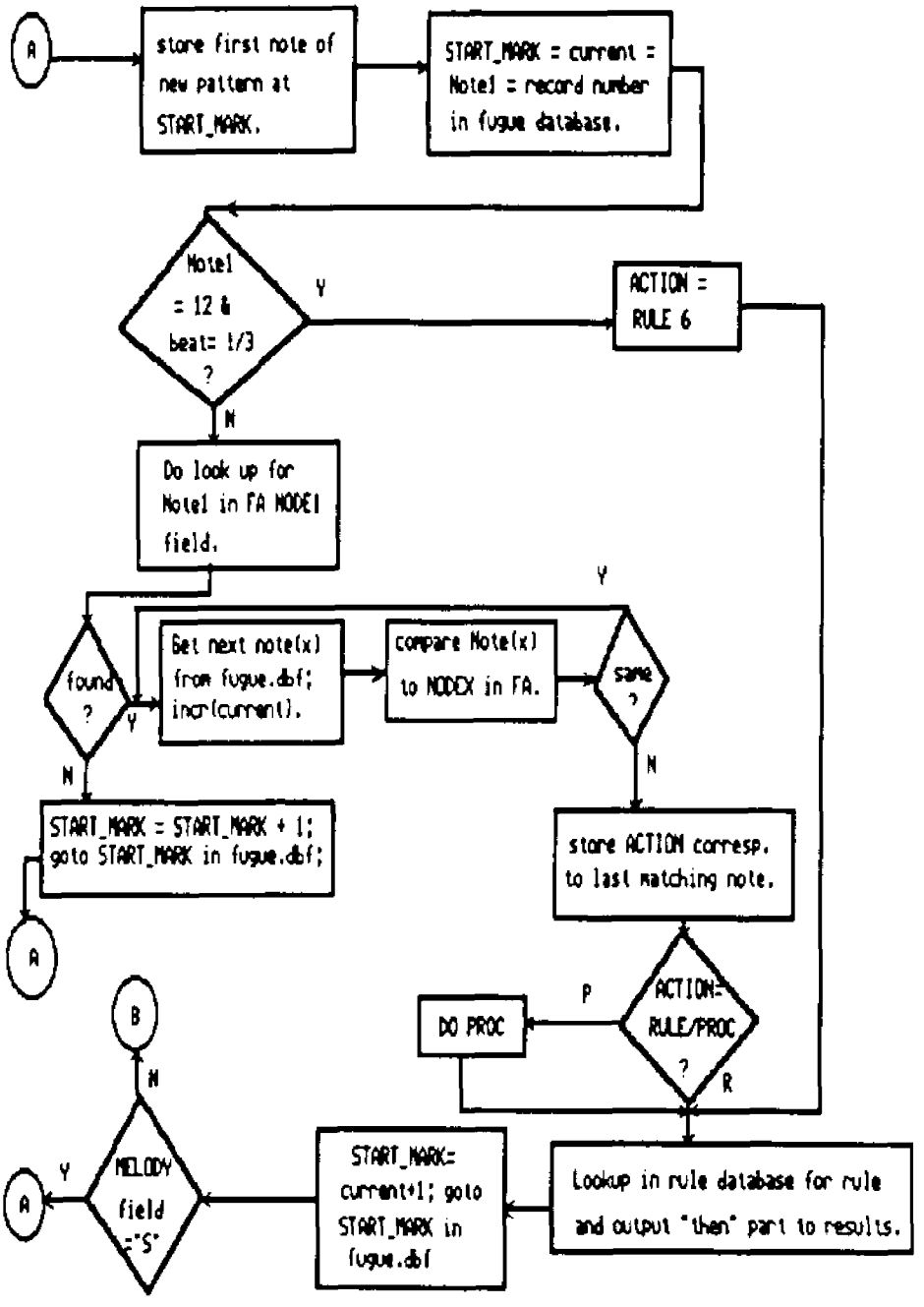
PROC6: Decides if RULE21 applies to an uneven sequence of 16ths.

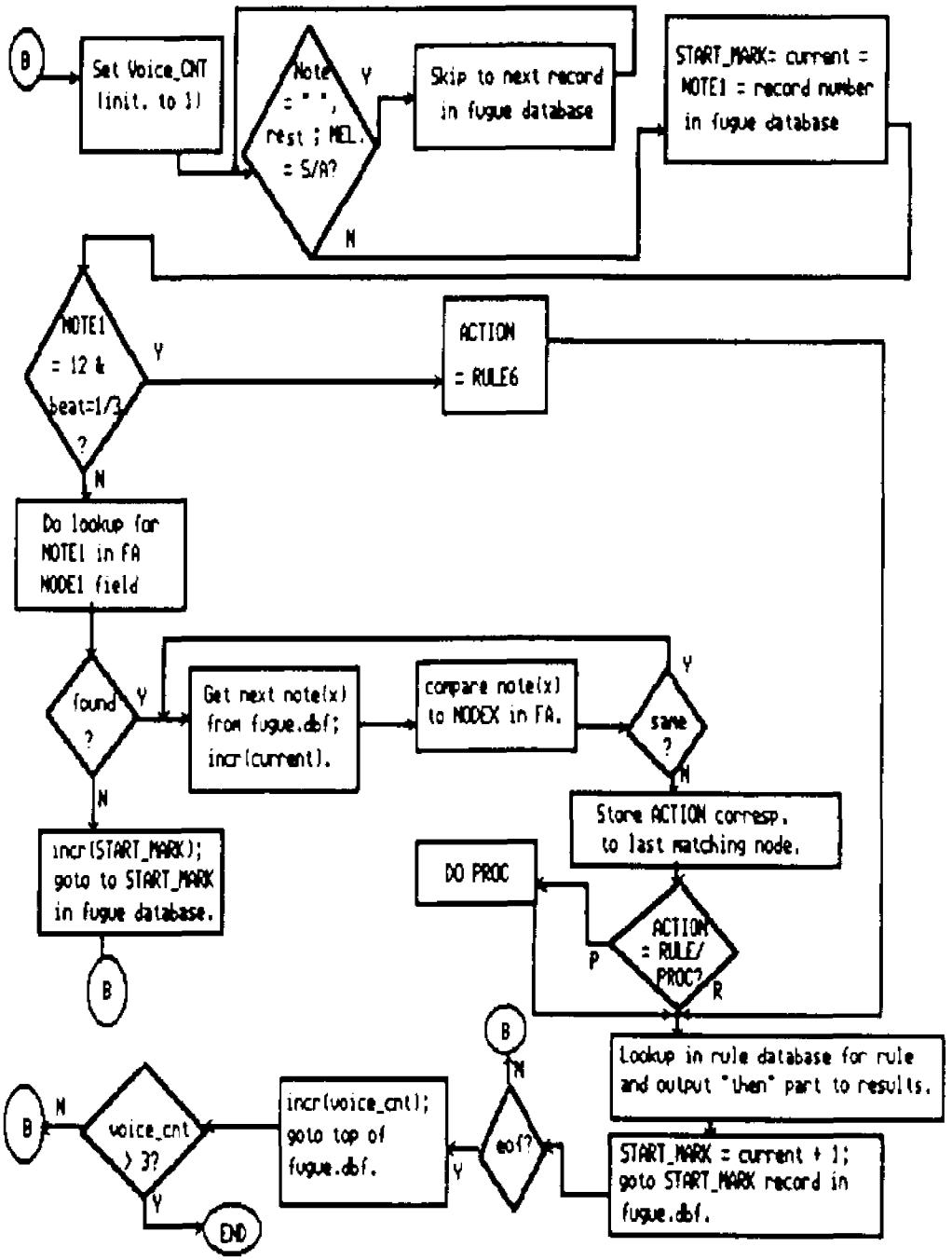
PROC7: Decides if RULE7 applies to a single 8th.

The following flowchart illustrates the design of the process program. The source code is listed in Appendix G. The output of the program is listed in Appendix F.

FIGURE 15: PROCESS.PRG Flowchart




A
store first note of new pattern at START_MARK.
START_MARK = current = Note1 = record number in fugue database.
Note1 = 12 & beat= 1/3 ?
Y
ACTION = RULE 6
N
Do look up for Note1 in FA NODE1 field.
found ?
Y
N
Get next note(x) from fugue.dbf; incr(current).
compare Note(x) to NODEX in FA.
same ?
Y
N
START_MARK = START_MARK + 1; goto START_MARK in fugue.dbf;
A
store ACTION corresp. to last matching note.
ACTION= RULE/PROC ?
P
R
DO PROC
B
N
MELODY field ="S"
Y
A
START_MARK= current+1; goto START_MARK in fugue.dbf
Lookup in rule database for rule and output "then" part to results.


B
Set Voice_CNT (init. to 1)
Note = " ", rest ; MEL. = S/A?
Y
N
Skip to next record in fugue database
START_MARK= current = NOTE1 = record number in fugue database
NOTE1 = 12 & beat=1/3 ?
Y
N
ACTION = RULE6
Do lookup for NOTE1 in FA NODE1 field
found ?
Y
N
Get next note(x) from fugue.dbf; incr(current).
compare note(x) to NODEX in FA.
same ?
Y
N
Store ACTION corresp. to last matching node.
incr(START_MARK); goto to START_MARK in fugue database.
B
DO PROC
ACTION = RULE/ PROC?
P
R
B
N
Lookup in rule database for rule and output "then" part to results.
START_MARK = current + 1; goto START_MARK record in fugue.dbf.
eof?
Y
incr(voice_cnt); goto top of fugue.dbf.
B
N
voice_cnt > 3?
Y
END

5.3.4. Validation

To validate the system implemented in this project, we compared the output listing as generated by the expert system, to the edited version by Anthony Newman. The edited music for a fugue from the subset is in Appendix D; the output listing is in Appendix F.

We found that the expert system generated the same editing instructions approximately 85 - 90% of the time, as tested on six fugues in the subset. Validation of the aural output from Music Printer Plus is outside the scope of this project.

5.4. Implementation Issues

What the expert system really does is provide an edition of a Bach fugue that represents Anthony Newman's interpretation. We are also providing aural musical examples for a future validation project. It is the aural examples that we discuss in this section.

5.4.1. Music Printer Plus

First, we consider the problems concerning the generation of aural output. This pertains to Music Printer Plus and the exactness of its interpretation of performance symbols. The table given in Appendix E defines exact values, e.g.

24 clicks per quarter note. According to the table, each note is played for the maximum value of the note which is a style of playing that a human performer cannot match. It is impossible for anyone to play a new note at precisely the moment the previous note was released. So, these time values are not going to sound "right". Perhaps we should subtract one or two clicks from each value to simulate a more realistic performing situation. How many should be subtracted? Maybe all quarter notes should be held for 22 clicks, and the extra two should be silence to produce an effect of a performer releasing and finding the new note. Maybe it should randomly be one sometimes, and two at other times. What if a note is to be played staccato? Then perhaps it should be held for 19 clicks.

Music Printer Plus handles these questions for us now; we put a staccato dot on a note in the screen notation and the compiler decides how many clicks to shorten it. Also, each note is held for exactly the correct number of clicks (24 per quarter) which, as stated earlier, cannot be done by a human performer.

The defaults of Music Printer Plus are arbitrary, but they are all we have at this point. Eventually, we can develop automatic interfaces which allow us to modify data at the MIDI message level. Then, we can assign realistic default values based on what our expert says after actually hearing

the output of the automated performer. The expert might say: "Those quarter notes should be shorter or longer within the framework of the 24 clicks." Or, what is more likely, a whole set of rules for just setting the default of each rhythmic value based on the characteristics of the piece, will have to be developed.

Still, we are coming up with something substantial using Music Printer Plus. The output, however, is not as expressive as it could be, had we more control over each MIDI message. It sounds too perfect, e.g., each staccato note sounds exactly like every other staccato note. In a sense, it is a perfect performance.

Another aspect of this problem is the fact that we cannot have metronome-perfect beats if we are to have expressive performance. No human can perform in perfect time, but a synthesizer can. We do need a consistent, but not perfect beat, if we want to simulate human performance. How "imperfect" should we make it? Again, we have the problem of trying to incorporate human imperfections to make the music sound more expressive.

5.4.2. Transition Graph

The second implementation issue is that the transition graph program finds every sequence where a rule or procedure

applies, which is what it should do. The action listing in Appendix F gives every possible thing a performer can do to interpret this fugue. But, a human performer would not do it all. He/she may apply a rule in certain places and not in others basing the decision not on a rule, but on what sounds right at the moment.

This problem has already been discussed in Appendix B. Bob James commented on the fact that general interpretation rules should apply, but they can be overridden by personal taste. A particular rule may apply in a situation, but if he prefers it to sound another way, he does not apply the rule. Such personal decisions cannot be formalized, and therefore cannot be built into the knowledge base. Unfortunately, "breaking the rules" in this manner creates a much more exciting performance because the expected does not happen.

These are all difficult implementation problems. We are trying to take something spontaneous and inexact, make it precise so the computer can handle it, but implement it in such a way that it sounds spontaneous and inexact. As stated earlier, we cannot make the final implementation sound too spontaneous using Music Printer Plus. When it comes time to develop the automatic interfaces, however, we will have the opportunity to do a lot of fine tuning. Only when we can directly modify MIDI messages will we have the control to create a more human-sounding expressive performance. But even

then, we still have the task of formalizing some very elusive rules based on personal taste and real-time processing.

5.5. Language-Theoretic Implications

Is the language processed by this system context-free or context-sensitive?

The actual operation of the transition graph on rhythmic patterns is context-free in that we are looking for patterns regardless of context and seeing if we can get anywhere in the graph. If we land on a final state with a rule, then the rule is based on the rhythmic pattern only. If we land on a final state with a procedure, context-sensitive melodic information is required to determine which rule applies.

Therefore, we are dealing with a context-free language representation in terms of the rhythmic patterns. However, when we require additional melodic information, the language becomes context-sensitive. Music itself is not a context-free language. We discussed in an earlier section how parts of music can be represented in a context-free manner. However, if we want to add meaning to the music through expressive performance, we must add context-sensitive material.

6. Future Directions and Uses

6.1. Additional Validation

The validation done for this project is sufficient to prove that we have an expert system that can edit a fugue the way our expert does. Whether the musical output of Music Printer Plus sounds like our expert performing is an additional validation exercise. Validating to this extent, would require the cooperation of Anthony Newman (and others), and would also require some way of modifying the defaults in Music Printer Plus.

Listening to music is a very subjective activity. To have Newman or other experts on his style, listen to our output and judge how much it sounds like the real thing would be a very difficult thing to measure.

6.2. Enhancements to the Expert System

What must be added to this system to include the other fugues of the Well-Tempered Clavier? Other Fugal works by Bach? Fugues by other composers? What about the non-fugal works?

The Well-Tempered Clavier has 2, 3, 4, and 5 voice fugues. Obviously, the databases and transition graph files must be enhanced to accommodate four and five voice

fugues. They are currently hard-coded to deal only with two or three voice fugues.

One or two more NOTE, RNOTE, STEP and MELODY fields must be added, and the programs enhanced to deal with more voices. These are minor changes; the real problem with adding extra voices is hardware-related. Because of the number of arrays required in the analyze program, we ran out of memory with large 3-voice fugues. Therefore, some method for going beyond the 640K limit of DOS must be implemented to do four or five voice fugues. Porting the code to OS-2 is one option; another is using memory managers like QEMM or the All Charge Card; or, rewriting the code on a mainframe.

The other problem with adding fugues from the Well-Tempered Clavier is the variety of meters used. Anthony Newman bases a lot of his specific rules on meter-related aspects. So, new transition graphs and rule sets would have to be elicited for each meter. The programs themselves would only require minor modifications such as opening the correct transition graph and rule files based on the meter passed in as a parameter. An interesting project would be to see what similarities exist between the rule sets for all the fugues.

These enhancements should be sufficient to handle all the fugues in the Well-Tempered Clavier, as well as most of the other fugues by Bach. As for other composers, if the composer is a contemporary of Bach and from the same Lutheran tradition, the system should work as is. However, a polyphonic work by an Italian or French composer would require different transition graphs and rule sets. The same holds true for non-fugal works, although these would require a different analyze program.

Basically, therefore, we tried to modularize the system so a new set of fugues could be added by adding a new transition graph and rule file. Non-fugal works would require program changes.

6.3. Automation of the Process

- What must be added to make the whole process automatic?

As discussed earlier, we first need a translator to translate the initial MIDI file into the numeric representation that the transition graph program uses. Then, we would run the transition graph program.

The output of the system will not be a list of instructions for use with Music Printer Plus. Instead, the initial MIDI

file will be modified directly. Rhythmic clicks will be added and subtracted; additional messages may be added for ornaments; etc. Therefore, a program must be developed that takes the action specified by the knowledge base and translates it to a physical change to the MIDI messages. This would require knowledge of the format of Music Printer Plus files. It would also require technical knowledge of the MIDI specification and a language (such as C) to modify the messages at the byte level.

The knowledge base must be expanded to handle the default values for each note and each type of performance action. Earlier, we discussed how to add certain "imperfections" to produce a more human-sounding performance. For example, a quarter note might be 23 clicks in one instance (1 click of silence), and 22 clicks in another instance (2 clicks of silence). The silence in between is to simulate the actual physical movement required by a human performer to get from one note to the other on the keyboard. A staccato note may be 19 clicks at one point and 21 some place else. We also need to create a consistent (but not perfect) beat, as well as have the ability to stretch and contract the beat in places. Adding all this spontaneity and randomness is a big job, requiring a great deal input from the expert himself.

Once all this is done, the only requirement will be to

input the notation into Music Printer Plus and compile it. A translator program is then run which translates the input to the numeric representation. Then, the program that applies the knowledge base is run, in conjunction with the program that translates the actions specified by the knowledge base into modifications to the MIDI file. The final output is a new MIDI file all ready to be played by the synthesizer.

6.4. Educational Uses

- A music student can take the notation of any piece and hear it performed in a number of different styles. For example, what if we apply Anthony Newman's style to a jazz piece or a Beethoven sonata? How would a Bach fugue sound played in an impressionistic style? How does Anthony Newman's style differ from Igor Kipnis? This could be very helpful to a student trying to learn the differences between performance styles of different players or between periods of music.

7. Conclusions

7.1. The Knowledge Structure Paradigm

Musical performance is not a typical application for an expert system. The fact that Orchard and Tausner's knowledge engineering methodology and Klir's knowledge structure formalization worked for such an unusual project, says a lot about the flexibility of the approach. The knowledge structure formalization provided an excellent paradigm for organizing the knowledge, and worked well as a framework for a design.

7.2. A "Human-Sounding" Performance

To create a "human-sounding" expert performance system, we need to incorporate human imperfection and spontaneity. Is this really possible?

We learned that there is only so far one can go in incorporating real "human-sounding" characteristics with a computer. Especially in this project, with our dependence on Music Printer Plus defaults, we could do very little beyond applying rules and coming up with a perfect "machine-sounding" application of those rules.

But it is possible to automate the interfaces and directly

modify MIDI messages, and eventually, this will be done. Then, how do we incorporate human spontaneity? Unfortunately, we will be faced with the monumental task of eliciting knowledge from experts that is even more elusive than the knowledge elicited for this thesis.

In fact, there may be some doubt that such knowledge can even be formalized. "What do you mean when you say 'This feels right' or 'This sounds better than this'?" We may end up with rules like: "If it is 9:00 on a Sunday night and I have just had a nice dinner and a couple glasses of wine, I play it this way; else, I play it the other way". We may have to develop transition graphs and rule sets based on mood, or time of day, all of which must be passed to the program as parameters.

Now, we have to stop and ask: What are we really doing here? Do we need to go so far with this? What would be the purpose of creating a program capable of basing decisions on human emotions and moods?

The value of doing it depends on the purpose. If we want a truly expressive "human-sounding" performance to enjoy the music, we should probably have a human do it. If we want to study the intricacies of human emotion, create lots of transition graphs and rule sets for educational purposes, or

if we are in the mood for a "perfect" performance, we have good reasons to proceed.

Whatever the value, the task itself would not be easy. It is very difficult to get a musician who is accustomed to performing instinctively, to slow down and become fully aware of his/her mental processes, especially when those mental processes concern emotion. Not to mention the fact that the input, transition graphs and rule sets will all grow exponentially to handle the additional data. As computers become more powerful, however, the prospects for continuation of this project become more positive, despite the considerable challenges involved.

Appendix A: MIDI

MIDI, the Musical Instrument Digital Interface, is a standardized set of instructions that can be communicated over a dedicated electrical line. The MIDI specification calls for various binary codes to be defined as musical functions such as "note on" (press a key), or "note off" (release a key). It also defines the electrical characteristics and data rate (31.25 kbaud) of the line used to carry the information.

Thus, MIDI was established as a hardware and software specification which would make it possible to exchange information between different musical instruments or other devices such as sequencers, computers or mixers. It was publicly announced in June 1983 after lengthy negotiations between Japanese and American synthesizer makers.

MIDI communication is achieved through multi-byte "messages" consisting of one status byte followed by one or two data bytes (certain message types have a different format).

A MIDI-equipped instrument typically contains a receiver and a transmitter. A receiver accepts messages in MIDI format and executes MIDI commands. A transmitter originates messages n MIDI format, and transmits them by way of a UART (Universal Asynchronous Receiver/Transmitter) and a line driver.

MIDI messages are sent over any of 16 channels which are used for a variety of performance information. One of the channels is called the "Basic Channel" over which it receives its main instructions such as program number or mode (mode refers to polyphonic or monophonic performance among other things). MIDI messages are divided into two categories: Channel and System.

There are five major types of MIDI messages: Channel Voice, Channel Mode, System Common, System Real-Time, System Exclusive.

Channel Voice messages control an instrument's voices which are the sequence of bytes sent over a channel. Channel Mode messages define an instrument's response to a voice message. System Common are messages sent to all receivers in the system, regardless of channel. System Real-Time messages are used for synchronization and are used by all clock-based units in a system. System Exclusive messages contain system messages that are exclusive to a particular manufacturer ID number that is part of the message.

Our main concern is channel voice messages, since these carry the actual melodic and rhythmic information. A channel voice message uses four bits in the status byte which addresses the message to one of the 16 channels. The other 4 bits of data in the status byte is used to define the message type.

The one or two data byte(s) that follow have different meanings depending on the message type. The following table defines channel voice messages.

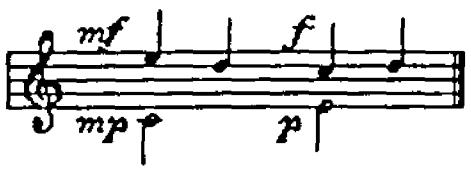
Channel Voice Messages

STATUS	DATA BYTES	DESCRIPTION
1000nnnn	0kkkkkkk 0vvvvvvv	Note off vvvvvvv: note off velocity
1001nnnn	0kkkkkkk 0vvvvvvv	Note on vvvvvvv <> 0: vel. vvvvvvv = 0: note off
1010nnnn	0kkkkkkk 0vvvvvvv	Polyphonic key pressure or Aftertouch vvvvvvv: pressure value
1011nnnn	0ccccccc 0vvvvvvv	Control Change (see below) ccccccc: control # (0-120) vvvvvvv: control value
1100nnnn	0ppppppp	Program change ppppppp: program # (0-127)
1101nnnn	0vvvvvvv	Channel pressure (Aftertouch) vvvvvvv: pressure value
1111nnnn	0vvvvvvv 0vvvvvvv	Pitch Bend Change LSB Pitch Bend Change MSB

Notes to Table:

1. nnnn = Voice Channel number (1-16) e.g., 0000 is channel 1, 0001 is channel 2 and 1111 is channel 16.
2. kkkkkkk: note number (0-127)
3. vvvvvvv: key velocity; this corresponds to how quickly the musician removes his or her finger from the key. A velocity value of 0 is interpreted as a note-off command.
4. Polyphonic key aftertouch is usually not implemented on most keyboards. It requires that each key have an attached pressure transducer, so that a musician can hold down a chord, adjust the pressure applied to each note with his or her fingers, and get a different aural response for each adjustment.
5. Control change commands are a catch-all for everything not defined elsewhere. The two bytes attached to a controller command defines, first, which controller is being used, and second, a value for the position of the controller. A controller in terms of MIDI instruments are things like a modulation wheel, breath controller, volume pedal, damper pedal, etc.
6. Progam change commands use a single byte to define a program number that an attached device is supposed to change.
7. Channel aftertouch is a value representing the current pressure applied to the keyboard.
8. Pitch bend is a controller that got its own command because of its popular use. The two bytes (least and most significant) indicate the amount of pitch bend.

Basically what happens is a series of bytes are sent over channels. A note-on byte is followed by a note value. This note will continue to sound until a note-off (or note-on with velocity = 0) is sent. The following brief example should help to illustrate: (Note: the time is in increments of 24 because there are 24 "clicks" per quarter note.)



This is how the above musical passage would be represented as a MIDI File. MIDI Files are an emerging standard for exchanging musical data among different programs and different computers. All numbers are decimal.

Time	Event Code	Other Bytes	Explanation
0	144	76 70	Beginning, Note On, Channel 1, pitch E3, velocity 70
0	145	60 58	Beginning, Note On, Channel 2, pitch C2, velocity 58
24	128	76 64	2nd quarter, Note Off, Channel 1, pitch E3 (default velocity)
24	144	74 70	2nd quarter, Note On, Channel 1, pitch D3, velocity 70
48	128	74 64	3rd quarter, Note Off, Channel 1, pitch D3
48	129	60 64	3rd quarter, Note Off, Channel 2, pitch C2
48	144	72 70	3rd quarter, Note On, Channel 1, pitch C3, velocity 96
48	145	64 48	3rd quarter, Note On, Channel 2, pitch E2, velocity 48
72	128	72 64	4th quarter, Note Off, Channel 1, pitch C3, velocity 64
72	144	74 70	4th quarter, Note On, Channel 1, pitch D3, velocity 96
96	128	74 64	5th quarter, Note Off, Channel 1, pitch D3
96	129	64 64	5th quarter, Note Off, Channel 2, pitch E2

Appendix B: Elicitation Techniques

Two professional musicians were used as the experts in this project: Bob James and Anthony Newman. Bob James is a prominent jazz composer and performer. Anthony Newman is a well-known keyboard performer.

Bob James was used as the expert on the interpretation decision-making process. The interview began with a description of exactly what we wanted to elicit from him, i.e., we wanted to help him to become more aware of a mostly unconscious activity. To do this, we showed him several pieces of keyboard or vocal music of vastly different styles and period. The pieces had no markings to indicate the composer or the name of the piece.

Then, we asked him leading questions. It was emphasized that he should try to remember his first impression on seeing the piece. This was used as a cue to gauge questions.

We got through four pieces: a Dona Nobis Pacem from the 17th century; an early Haydn piano sonata; a Mendelssohn organ sonata; and, Fugue #2 from the Well-Tempered Clavier I. In between the Mendelssohn and Bach, he was shown a print of a Breughel painting as a break. This seemed to be a welcome relief; it became clear that he could not have gone for more than 90 minutes, as the exercise was quite intense.

We learned the following from this interview:

1) His first impression usually pertained to a guess as to the composer and period. He related that once he had an idea of period, he knew how much freedom he could take in the interpretation. Music from more recent periods (19th and 20th century) are usually marked very thoroughly, whereas older music leaves more to the performer. He also made the point that listeners are more willing to accept more novel interpretations of older music than newer music. This is because older music is so much more familiar.

In addition to learning this from composer and period, he also got clues that helped him to decide on the general feel of the piece. These clues usually came from the musical structure. For example, he noticed right off that the Haydn piano sonata was melody and accompaniment, and the accompaniment was in a style typical of the early Classical period. This gave him many clues to the interpretation, e.g., bring out the melody and make the accompaniment softer. The Mendelssohn on the other hand, began with big chords and also had a marking of "Con moto maestoso". This told him the tempo should be somewhat slow and the instrument should have a very full sound.

Thus, he first tried to place the piece in music

history. Then, he looked at the general structure of the piece to either help him place it in history or for interpretation clues.

2) Usually after focusing on his first impression, he would get very quiet: he was singing the melody to himself. The first question here was how could he tell which part was the melody. His reply was that it is usually the top line and/or the most active line rhythmically. A lot of ornamentation on a line is another clue to melody.

Why was finding the melody so important? He replied that this tells him where the phrases are, and helps him to see the overall structure of the piece.

We spent a lot of time discussing how to tell the end of the melody which is usually a cadence. Knowing where cadences are is very important to making decisions about interpretation, because a cadence is the goal of the melody and harmony. His ideas on this were: if there are words, look at the breaks in the word phrases - these usually correspond to the musical breaks. Also, the last note of the phrase is longer in duration to indicate a point of repose. Another clue is to look at the line of the melody; it will usually

go up and then come down at a cadence point.

3) Another major point to cover with him was how to decide on a tempo. He again got clues for this from the overall structure of the piece. For example, the Haydn piano sonata was marked "Allegro" and this was supported by the nature of the melody. The melody had lots of ornaments which to him meant that it should be dance-like, so not too fast. The chordal structure of the Mendelssohn as well as the marking told him slow and strong. The Dona Nobis Pacem should be somewhat slow and lyrical because there were several notes to a syllable.

4) Finally, we asked if there are rules that he applies in deciding on an interpretation. He said there are rules such as "Emphasize the first beat of a measure" or "Slur very fast notes together". Such rules are important, but so is how a piece sounds. Sometimes, he might purposely not emphasize the first note of a measure (which according to a rule should be), because he personally likes the sound another way. He also commented that such personal decisions cannot be formalized, which means we cannot incorporate such "knowledge" into the system.

In general, this interview gave a much clearer idea as to the overall process of making interpretation decisions. In terms of this project on fugues, we translated Bob's approach into the following process in knowledge structures:

1) Start in structure to determine the structure of the piece. In our case, we already know this since this automated performer deals only with Bach fugues.
2) In structure, find the main melody (fugue subject) and see where it ends.
3) Go to behavior to decide on an interpretation of the fugue subject.
4) Continue, by processing more notes using the rules in behavior and the overall structure. Override the rules if it sounds better another way.

(Note: a tape of this interview is available.)

This general process was then filled in with details from interviews with Anthony Newman and research into his writings. The "interviews" were conducted 4 years ago during the course of my studying harpsichord with him. These lessons were taped, so we have lots of material on his approach to the interpretation of Bach keyboard works.

To get all this information into some kind of format that a program can use, we went through the following steps:

1) Read his book on "Bach and the Baroque" (Newman, 1985) which discusses all of his ideas on performance practice. Many general rules could be taken from this.

2) Study his edition of the Well-Tempered Clavier in detail, to learn exactly what articulations he believes are best. We were able to get some very detailed rules from this study.

3) Listen to the lessons to see if the rules from steps 1 and 2 held up, which they did. We also listened to his recordings to see if the rules were supported - they were.

Thus, the general decision-making strategy comes from Bob James, while the specific rules on how to interpret a Bach fugue come from Anthony Newman.

Appendix C: Musical Background

>>> Historical Background

- The Baroque Period

In music history, the Baroque period extends from about the end of the 16th century to the middle of the 18th century. During this time, certain ways of organizing musical materials, certain ideals of musical sounds and certain kinds of musical expression developed from diverse beginnings to a sophisticated system culminating in the works of Handel and Bach.

The music of the previous period was predominantly vocal, although instrumental music was becoming more and more popular in the 16th century. As the 17th century began, new forms of composition emerged including the opera, the cantata, and the oratorio in vocal music; and the concerto, sonata and several forms of solo keyboard music in instrumental music.

The main keyboard instruments of the Baroque period were organ and harpsichord. Most keyboard music of the era falls into one of the following categories:

1) Ricercare: pieces in continuous, non-sectional imitative counterpoint (similar sounding, independent melodies all playing simultaneously). These pieces were called ricercare, fantasia, capriccio, fuga,or verset; these eventually develop into the fugue.

2) Canzona: pieces in discontinuous, sectional imitative counterpoint, sometimes mixed up with other types of keybaord music.
3) Pieces based on a given melody or bass. Such compositions are called theme and variations, partita, passcaglia or chaconne, chorale partita or chorale prelude (these last two are based on Lutheran chorale melodies).
4) Dances based on popular dance rhythms of the day (e.g., allemande, gavotte, courente, gigue, etc.)
5) Pieces in an improvisatory style: toccata, fantasias or prelude.

Most keyboard music was written either for performance in the church, or at small court concerts.

- Bach

Bach's career was similar to that of many successful musicians of the time in Lutheran Germany. He was organist at Arnstadt (1703-07) and Mühlhausen (1707-08); court organist and later Concertmaster in the chapel of the Duke of Weimar (1708-17); Music Director at the court of a prince in Cöthen (1717-23); and Cantor of St. Thomas's school and Music Director in Leipzig (1723-50).

Bach had a reputation as an organ virtuoso and a writer of sophisticated polyphonic works, but there were several other contemporary composers more widely known in Europe. He regarded himself as a craftsman doing a job to the best of his ability for the satisfaction of his superiors, for the pleasure of his fellow men, and to the glory of God.

Bach was trained as a violinist and organist. His early works were written for organ since he was required to compose in his position at Arnstadt. It wasn't until he arrived in Weimar and later Cöthen, that he began to really compose secular keyboard music for harpsichord. This music would have been performed in small court concerts or musicales.

Bach quite often used numeric symbolism in his pieces. Sometimes the statements of a fugal subject might appear in measures that signify a special series (e.g., Fibonacci). Sometimes the highest note in a piece and therefore, a climactic point happens at exactly the middle of the piece.

- The Well-Tempered Clavier

The Well-Tempered Clavier Book I was completed in Cöthen in 1722. It consists of 24 preludes and fugues in each of the twelve major and minor keys. In addition to demonstrating the possibility (with the then novel idea of tempered tuning) of using all the keys, Bach had specific

didactic intentions. Each of the preludes presents a particular technical task to the player. The fugues are a compendium of all the possibilities of monothematic fugal writing.

>>> Performance Practice (Harpsichord in Particular)

- The Harpsichord

A harpsichord looks like a small piano and its basic "works", a set of tuned strings activated from a keyboard, are essentially those of a piano. The difference comes in the way the sound is generated. On a piano, the sound is produced by striking strings with soft, felt-covered hammers which are controlled by a complicated mechanism called the action. The complication arises from the need to permit slight changes of finger pressure to achieve all possible dynamic shadings, from very soft to very loud, and then have the sound stopped, or damped when the fingers are lifted off.

On the harpsichord, sound is produced by plucking the strings with small, hard quills, one per string, activated from the keyboard. Harpischord action and damping are simple and direct, and thus do not allow for the subtle changes in dynamics that are possible on a piano. This mechanism provides for one level of loudness which can only be modified by adding more strings to be plucked.

There are many implications of this action to performance. First, without the possibility of dynamic variation, the expressiveness of a harpsichord is far more subtle than piano. Anyone listening to harpsichord music for the first time typically finds it monotonous and difficult to understand. It takes time to learn the subtleties of harpsichord technique. Another implication is the sound from the plucked string dies away very quickly because it is damped, so techniques were developed to simulate a continuing sound.

- Playing a Harpsichord

One of the most important goals in performing music is to shape the musical line as it flows through time. The first step in doing this is to sustain the flow by not distorting the basic pulse of the music; this gives us the framework within which to work. The second step is to inflect the flow of sound with phrasing, articulation, dynamics, rhythmic and other modifications which are used to mold it into expressive patterns.

According to Newman (1985), the harpsichordist has three variables to manipulate in order to expressively interpret a piece:

1) Rhythmic alteration
2) Articulation
3) Touch

Rhythmic alteration refers to the freedoms a performer might take in the treatment of the rhythmic flow of a piece. These alterations allow the harpsichordist to emphasize important parts such as cadences or climaxes. Many of the alterations are very subtle; for example, fringing (Newman, 1985) is a technique where the bass is played slightly before the upper voices.

Articulation refers to the degree of separation between successive notes. The middle ground is no separation, i.e., one sound stops exactly when the next sound starts. The extremes are from overlapping notes (playing new notes without letting up the keys of previously played notes) to distinct separation of successive notes. The range from one extreme to another provides interpretation choices for the harpsichordist.

Touch refers to the lightness or heaviness of the hand in playing. Although the loudness will not change regardless of how heavy or light one plays a harpsichord, the nature of the sound is different.

One of the basic concepts of Newman's approach to performance practice is the idea of strong/weak:

> "Strong/weak alternations underlie Baroque music at every structural level. As already stated, it is impossible to discuss meters, tempos, accents, fingering, rubato, or almost any other specific performance problem without being aware of the

constant presence of alternations between strong and weak, also called 'good/bad' or 'principal/ passing' metric units. This is true whether we are talking about individual notes as subdivisions of the measure, about beats as subdivisions of the measure or about measures and larger structural units."

The best description of the concept comes from a Baroque source (Quantz, 1752 as translated by Reilly, 1966):

"Here I must make a necessary comment about the length of time each note should be held. One must know how to make a difference in performance between the main notes, also called principal notes, or by the Italians, "good" notes, and the passing notes, also called by some foreigners, "bad" notes. Wherever possible, the principal notes must be brought out more strongly than the passing notes. In order to follow this rule, the fastest notes in every piece in a moderate tempo, or even in an Adagio, even though they appear to have the same value, must be played somewhat unevenly; thus the principal notes of each figure, that is, the first, third, fifth, seventh will be held a little longer than the passing notes, that is, the second, fourth, sixth

> and eighth; but this holding of the notes should not make as big a difference as the writing of dots beside them would."

Many of the interpretation rules in Newman's work concern the determination of strong/weak patterns, and the subsequent emphasis of the strong. His basic tenets are that Baroque music is never played strictly in time, and rhythmic freedoms are built on the premise of strong/weak units.

Strong/weak units within a measure are determined by the natural rhythmic emphasis of the figure. The beginning notes of a passage are always the strong ones, and the meter and tempo tell us the other strong ones. For example, in 4/4 time, the strong notes will be the ones that occur on beat 1 and beat 3. A chart is given above within the behavior knowledge structure that indicates what notes within a measure are strong based on meter and tempo. Strong notes should be stretched in time, i.e., held just a little longer than their written duration. If we could play them louder, we would do that too, but that is not an option on a harpsichord.

In addition to recognizing strong/weak within a measure, we also need to determine these alterations at higher structural levels, and articulate these as well. In general, an entire measure is strong if:

1) the theme begins on a downbeat (the downbeat is the first beat of the measure).
2) there is a cadence in any part of the measure.
3) there is a chord on the downbeat in a thin texture.
4) the texture of a measure is thicker than that of measures preceding it.
5) the downbeat has an ornament.
6) there is a large leap after the note on the downbeat or third beat.
7) there is dissonance.

A measure is weak if:

1) there is a tie from a previous measure onto or over the downbeat.
2) there is a series of sequences.
3) if the first measure material begins with a rest.

The ways to highlight a strong measure or strong beat within a measure are to prolong the downbeat (or another strong beat) slightly, perhaps the value of an additional 32nd note. The time that is added to make this accent is not subtracted elsewhere in the measure. Another way to accentuate a strong measure is to ornament the downbeat chord. Ornaments can also be used to emphasize a strong beat in the measure. Fringing, as defined above, is still another method.

Within the measure, as Quantz indicated, it is correct to play evenly written notes as if they were slightly dotted. The amount of inequality is a matter of taste.

Before leaving the topic of rhythmic alteration, we need to define rubato. Rubato is a flexible, elastic tempo involving slight speeding-up and slowing-down according to the requirements of musical expression. Sometimes the time "robbed" in one place is given back later in the measure, sometimes not. The techniques described above give a piece rubato, as well as the general rule of leading up to a cadence by a slight slowing down of tempo.

Ornamentation and embellishment are an important part of Baroque performance. In general, ornaments are usually added to punctuate a phrase (emphasizing the strong), while embellishments are decorations originally improvised by the performer. In terms of the fugue, a general rule is any ornament added to a fugue subject should be carried through to all successive statements of the subject. We don't have to use exactly the same ornament each time, but the ornamented area of the subject should be ornamented throughout the fugue.

The principles governing all rhythmic alterations and ornamentation are subject to personal interpretation. The performer should never allow the latitude of rhythmic alteration or ornaments to alter the predictable feeling

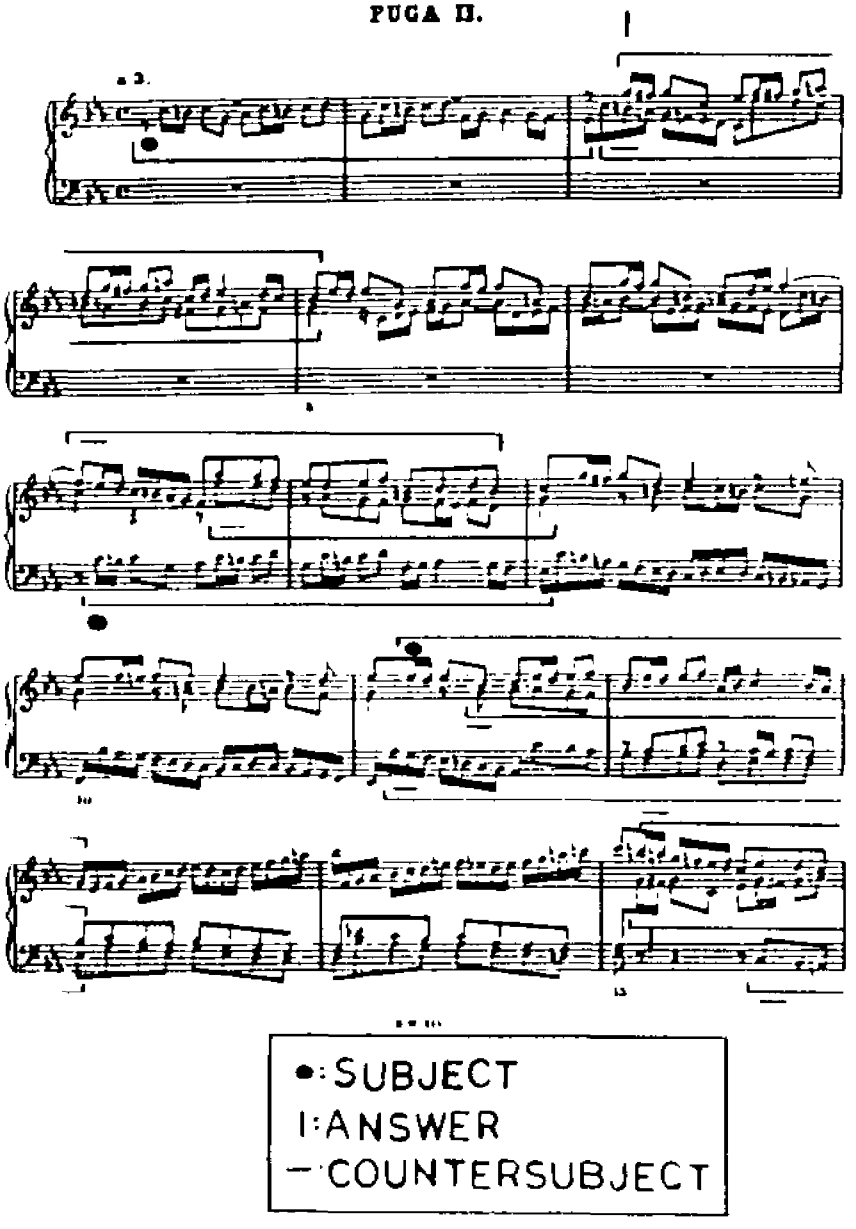
of the basic beat.

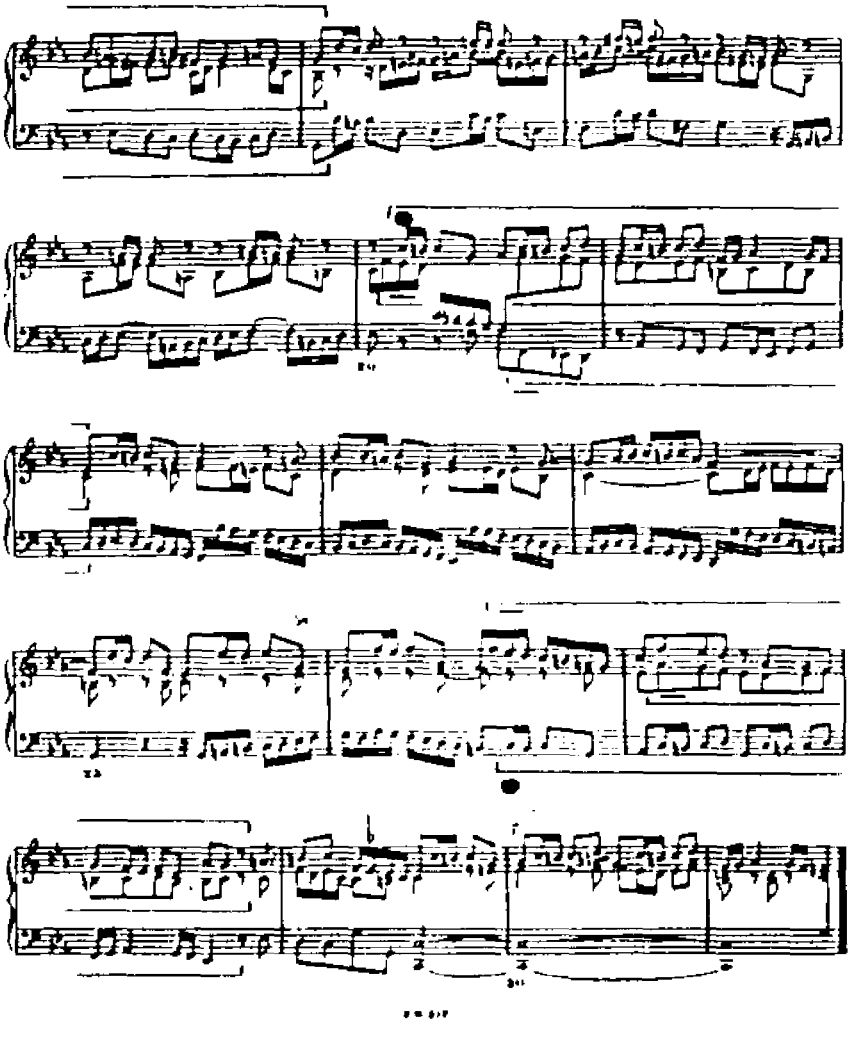
>>> Musical Form

Musical form involves the overall structuring and organization of the flow of musical time. A form is typically a plan for the composer and provides a means for him to balance unity and variety in a piece of music.

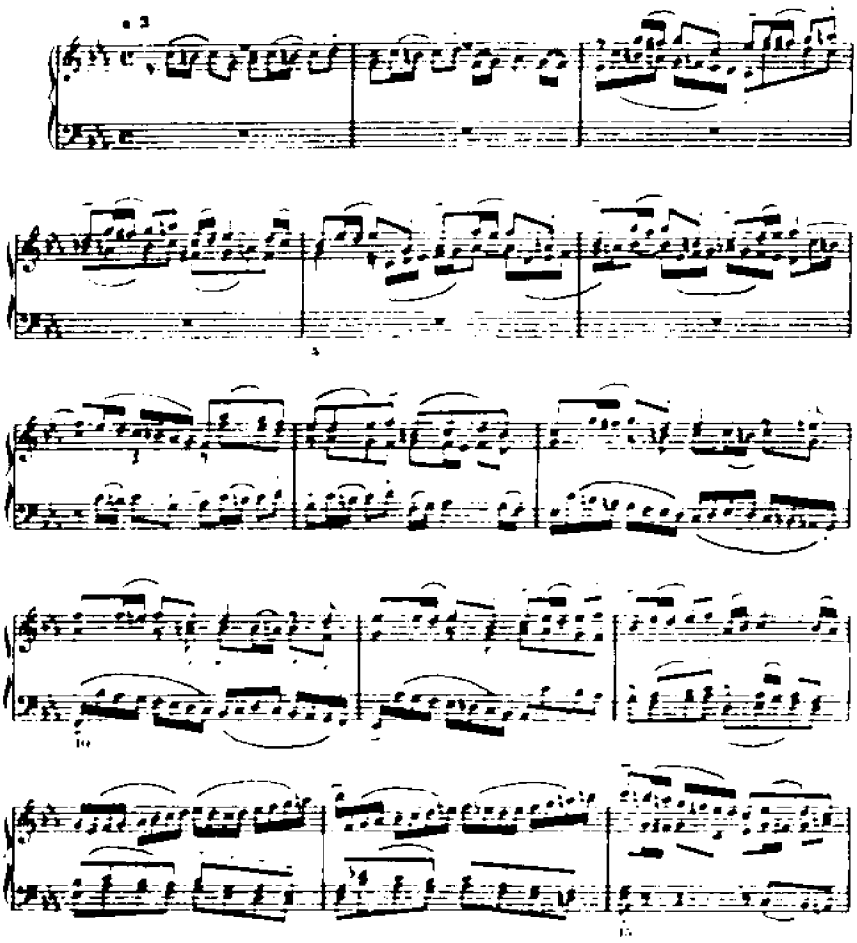
Some of the more common musical forms are: 1) Binary Form consists of 2 contrasting musical sections, A and B. Each section is usually repeated resulting in the pattern AABB. The A section begins in the tonic key and usually modulated to the dominant (V) key. The B section begins in the dominant and returns to the tonic. 2) Ternary Form has the structure ABA. The A section is usually entirely tonic and the B section is in some contrasting key. 3) Alternating Forms consist of a main section (A) which alternates with contrasting sections (B, C, D, etc.) in contrasting keys. There can be several sections alternating but typically the number of sections is three, five or seven. 4) Variation Form begins with a theme which is stated several times, but changed or varied in some particular way each time it is stated. There are typically modulations from variation to variation.

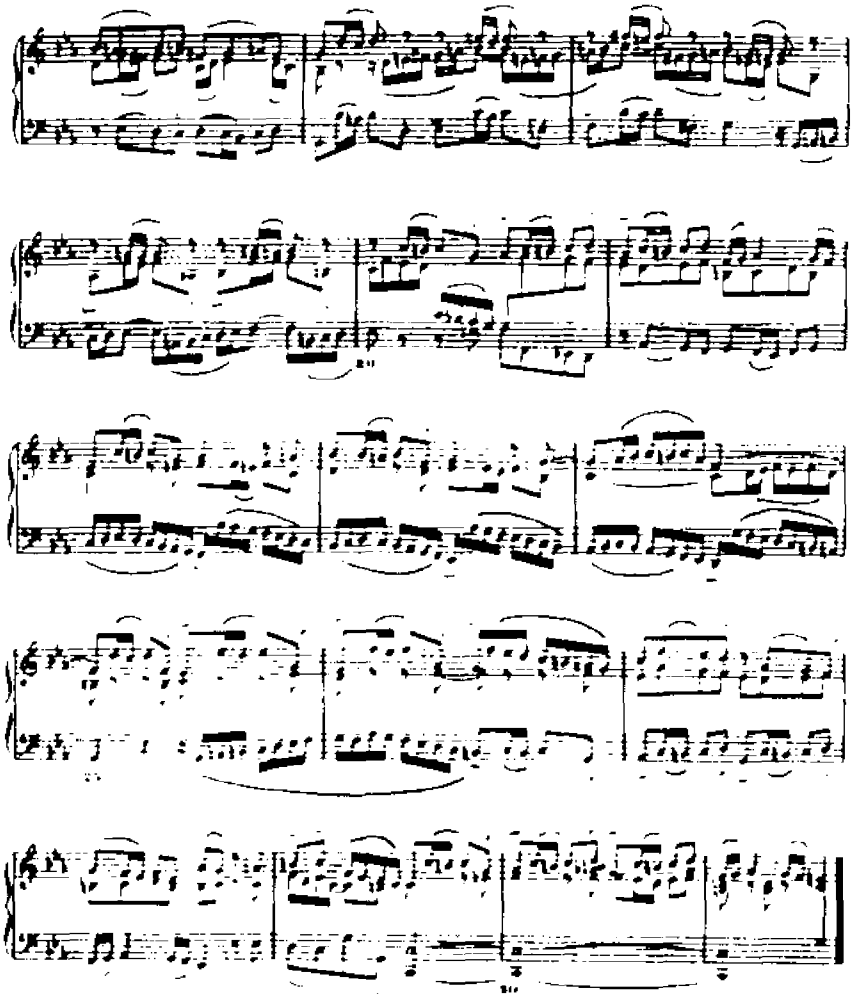
Appendix D: Fugue #2 in C minor (WTC I)





FUGA II.





Appendix E: Numeric Representation of Fugue #2 in C minor (WTC I)

BAR	BT	NOTE1	RNOTE1	NOTE2	RNOTE2	NOTE3	RNOTE3	MEL1	MEL2	MEL3
1	1	9999	96	9999	12	9999	96			
				0	6				S	
				-1	6				S	
	2			0	12				S	
									S	
				-5	12				S	
									S	
	3			-4	12				S	
									S	
				0	6				S	
				-1	6				S	
	4			0	12				S	
									S	
				2	12				S	
									S	
2	1	9999	96	-5	12	9999	96		S	
									S	
				0	6				S	
				-1	6				S	
	2			0	12				S	
									S	
				2	12				S	
									S	
	3			-7	6				S	
				-5	6				S	
				-4	24				S	
									S	
	4								S	
									S	
				-5	6				S	
				-7	6				S	
3	1	9999	12	-9	6	9999	96		S	
				0	6					
		0	6	-1	6			A	CS	
		-1	6	-3	6			A	CS	
	2	0	12	-5	6			A	CS	
				-7	6			A	CS	
		-7	12	-9	6			A	CS	
				-10	6			A	CS	
	3	-4	12	-12	12			A	CS	
								A	CS	
		0	6	3	12			A	CS	
		-1	6					A	CS	
	4	0	12	2	12			A	CS	
								A	CS	
		2	12	0	12			A	CS	
								A	CS	
4	1	-5	12	-2	12	9999	96	A	CS	

								A	CS
		0	6	-3	12			A	CS
		-1	6					A	CS
	2	0	12	-2	12			A	CS
								A	CS
		2	12	0	12			A	CS
								A	CS
	3	-7	6	-6	12			A	CS
		-5	6					A	CS
		-4	24	-5	12			A	CS
								A	CS
	4			-3	12			A	CS
								A	CS
		-5	6	-6	12			A	CS
		-7	6					A	CS
5	1	-9	12	-5	24	9999	96	A	CS
		-4	6						
		-5	6						
	2	-4	12	9999	6				
				-12	6				
		-12	12	-10	6				
				-9	6				
	3	-11	12	-7	6				
				-5	6				
		-2	6	-4	18				
		-4	6						
	4	-2	12						
				-10	6				
		-10	12	-9	6				
				-7	6				
6	1	-9	12	-5	6	9999	96		
				-3	6				
		0	6	-2	18				
		-2	6						
	2	0	12						
				-9	6				
		-8	12	-7	6				
				-5	6				
	3	-7	12	-4	6				
				-5	6				
		-5	6	-7	6				
		-4	6	-9	6				
	4	-2	36	-10	12				
				0	6				
				-1	6				
7	1			0	24	9999	12		
		-4	6			0	6	CS	S
		-5	6			-1	6	CS	S
	2	-7	6	9999	24	0	12	CS	S
		-9	6					CS	S
		-11	6			-5	12	CS	S

		-12	6					CS	S
	3	-14	12	9999	12	-4	12	CS	S
								CS	S
		1	12	5	12	0	6	CS	S
						-1	6	CS	S
	4	0	12	4	12	0	12	CS	S
								CS	S
		-2	12	2	12	2	12	CS	S
								CS	S
8	1	-4	12	9999	12	-5	12	CS	S
								CS	S
		-5	12	-4	12	0	6	CS	S
						-1	6	CS	S
	2	-4	12	-5	12	0	12	CS	S
								CS	S
		-2	12	-7	12	2	12	CS	S
								CS	S
	3	-8	12	-5	12	-7	6	CS	S
						-5	6	CS	S
		-7	12	-7	6	-4	24	CS	S
				-9	6			CS	S
	4	-5	12	-7	12			CS	S
								CS	S
		-9	12	-10	12	-5	6	CS	S
						-7	6	CS	S
9	1	-7	12	-5	24	-9	6	CS	S
						0	6		
		0	6			-1	6		
		-1	6			-3	6		
	2	0	12	9999	12	-5	6		
						-7	6		
		-5	12	-1	12	-9	6		
						-10	6		
	3	-4	24	0	12	-12	6		
						-10	6		
				0	6	-9	6		
				-1	6	-10	6		
	4	9999	12	0	12	-12	6		
						-14	6		
		-3	12	-5	12	-16	6		
						-17	6		
10	1	-2	12	-4	24	-19	6		
						-2	6		
		-2	6			-4	6		
		-3	6			-5	6		
	2	-2	12	9999	12	-7	6		
						-9	6		
		-7	12	-3	12	-10	6		
						-12	6		
	3	-5	24	-2	12	-14	6		
						-12	6		
				-2	6	-10	6		
				-3	6	-12	6		
	4	9999	12	-2	12	-14	6		

						-16	6		
		-5	12	-7	12	-17	6		
						-19	6		
11	1	-4	12	-5	24	-21	6		
						-4	6		
		0	6			-5	6	S	CS
		-1	6			-7	6	S	CS
	2	0	12	9999	12	-9	6	S	CS
						-11	6	S	CS
		-5	12	-5	12	-12	6	S	CS
						-14	6	S	CS
	3	-4	12	-4	12	-16	12	S	CS
								S	CS
		0	6	-4	12	0	12	S	CS
		-1	6					S	CS
	4	0	12	-5	12	-2	12	S	CS
								S	CS
		2	12	-7	12	-4	12	S	CS
								S	CS
12	1	-5	12	9999	12	-5	12	S	CS
								S	CS
		0	6	-16	12	-7	12	S	CS
		-1	6					S	CS
	2	0	12	-14	12	-5	12	S	CS
								S	CS
		2	12	-12	12	-4	12	S	CS
								S	CS
	3	-7	6	9999	12	-10	12	S	CS
		-5	6					S	CS
		-4	24	-16	6	-9	12	S	CS
				-17	6			S	CS
	4			-16	12	-7	12	S	CS
								S	CS
		-5	6	-19	12	-10	12	S	CS
		-7	6					S	CS
13	1	-9	6	-14	12	-9	12	S	CS
		-12	6						
		-10	6	-12	12	-4	12		
		-8	6						
	2	-7	6	-14	12	-5	12		
		-5	6						
		-3	6	-16	12	-7	12		
		-1	6						
	3	0	6	-14	12	-5	12		
		-1	6						
		-3	6	-17	12	-9	12		
		-1	6						
	4	0	6	-19	12	-10	12		
		2	6						
		4	6	-21	12	-12	12		
		6	6						
14	1	7	6	-19	12	-10	12		
		-10	6						
		-8	6	-11	12	-2	12		

		-7	6						
	2	-5	6	-12	12	-4	12		
		-3	6						
		-1	6	-14	12	-5	12		
		1	6						
	3	2	6	-12	12	-4	12		
		0	6						
		-1	6	-16	12	-7	12		
		0	6						
	4	2	6	-17	12	-9	12		
		4	6						
		6	6	-19	12	-10	12		
		8	6						
15	1	9	12	-17	12	-9	12		
		8	6	0	6	9999	12	CS	A
		6	6	-1	6			CS	A
	2	4	6	0	12	9999	24	CS	A
		2	6					CS	A
		0	6	-7	12			CS	A
		-1	6					CS	A
	3	-3	12	-4	12	9999	12	CS	A
								CS	A
		0	12	0	6	-12	12	CS	A
				-1	6			CS	A
	4	-1	12	0	12	-14	12	CS	A
								CS	A
		-3	12	2	12	-15	12	CS	A
								CS	A
16	1	-5	12	-5	12	9999	12	CS	A
								CS	A
		-6	12	0	6	-9	12	CS	A
				-1	6			CS	A
	2	-5	12	0	12	-10	12	CS	A
								CS	A
		-3	12	2	12	-12	12	CS	A
								CS	A
	3	-9	12	-7	6	-10	12	CS	A
				-5	6			CS	A
		-8	12	-4	24	-12	6	CS	A
						-14	6	CS	A
	4	-6	12			-12	12	CS	A
								CS	A
		-9	12	-5	6	-10	12	CS	A
				-7	6			CS	A
17	1	-8	12	-9	12	-17	12	CS	A
		-1	6	9999	12	-2	6		
		-3	6			-3	6		
	2	-1	12	9999	6	-2	12		
				-5	6				
		9999	12	-3	6	-10	12		
				-1	6				
	3	9999	12	0	6	-9	12		

				2	6				
		1	6	3	18	0	6		
		-1	6			-2	6		
	4	1	12			0	12		
				-3	6				
		9999	12	-2	6	-8	12		
				0	6				
18	1	9999	12	2	6	-7	12		
				3	6				
		3	6	5	18	2	6		
		1	6			0	6		
	2	3	12			2	12		
				-1	6				
		9999	12	0	6	-6	12		
				2	6				
	3	9999	12	3	12	-5	24		
		-8	6	-4	6				
		-10	6	-5	6				
	4	-8	12	-4	12	9999	6		
						-17	6		
		9999	12	-12	12	-15	6		
						-13	6		
19	1	9999	12	-11	12	-12	6		
						-10	6		
		-6	6	-2	6	-9	18		
		-8	6	-4	6				
	2	-6	12	-2	12				
						-15	6		
		9999	12	-10	12	-14	6		
						-12	6		
	3	9999	12	-9	12	-10	6		
						-9	6		
		-4	6	0	6	-7	18		
		-6	6	-2	6				
	4	-4	12	0	12				
						-13	6		
		9999	12	-8	12	-12	6		
						-10	6		
20	1	9999	12	-7	6	-9	12		
				-2	6				
		0	6	-4	6	9999	12	S	CS
		-1	6	-5	6			S	CS
	2	0	12	-7	6	9999	12	S	CS
				-9	6			S	CS
		-5	12	-11	6	-8	12	S	CS
				-12	6			S	CS
	3	-4	12	-14	12	-7	12	S	CS
								S	CS
		0	6	1	12	-19	12	S	CS
		-1	6					S	CS
	4	0	12	0	12	-21	12	S	CS
								S	CS
		2	12	-2	12	-22	12	S	CS

								S	CS
21	1	-5	12	-4	12	9999	12	S	CS
								S	CS
		0	6	-5	12	-16	12	S	CS
		-1	6					S	CS
	2	0	12	-4	12	-17	12	S	CS
								S	CS
		2	12	-2	12	-19	12	S	CS
								S	CS
	3	-7	6	-8	12	-17	12	S	CS
		-5	6					S	CS
		-4	24	-7	12	-19	6	S	CS
						-21	6	S	CS
	4			-5	12	-19	12	S	CS
								S	CS
		-5	6	-8	12	-17	12	S	CS
		-7	6					S	CS
22	1	-9	12	-7	24	-12	6	S	CS
						-10	6		
		0	6			-9	6		
		-1	6			-10	6		
	2	0	12	9999	12	-12	6		
						-14	6		
		-5	12	-3	12	-16	6		
						-17	6		
	3	-4	24	-2	12	-19	6		
						-2	6		
				-2	6	-4	6		
				-3	6	-5	6		
	4	9999	12	-2	12	-7	6		
						-9	6		
		-3	12	-7	12	-10	6		
						-12	6		
23	1	-2	12	-5	24	-14	6		
						-12	6		
		-2	6			-10	6		
		-3	6			-12	6		
	2	-2	12	9999	12	-14	6		
						-16	6		
		-7	12	-5	12	-17	6		
						-19	6		
	3	-5	24	-4	12	-21	6		
						-4	6		
				-4	6	-5	6		
				-5	6	-7	6		
	4	9999	12	-4	12	-9	6		
						-10	6		
		-5	24	-9	12	-12	6		
						-14	6		
24	1			-7	52	-16	6		
						-14	6		
		-4	6			-12	6		
		-2	6			-14	6		
	2	0	6			-16	6		

		-1	6			-17	6	
		0	6			-19	6	
		-4	6			-21	6	
	3	-7	52			-22	6	
						-5	6	
				-5	6	-7	6	
				-4	6	-9	6	
	4			-2	6	-10	6	
				-4	6	-12	6	
				-2	6	-13	6	
				-5	6	-15	6	
25	1			-8	12	-17	24	
		2	6	9999	12			
		0	6					
	2	2	12	9999	12	9999	24	
		-7	12	-8	12			
	3	-9	12	-7	12	9999	6	
						-17	6	
		3	6	9999	12	-15	6	
		2	6			-13	6	
	4	3	12	9999	12	-12	6	
						-10	6	
		-5	12	-3	12	-9	6	
						-7	6	
26	1	-7	12	-4	12	-5	6	
						-7	6	
		5	6	9999	12	-4	6	
		3	6			-5	6	
	2	5	12	9999	12	-7	6	
						-9	6	
		-4	12	-2	24	-10	6	
						-12	6	
	3	-5	6			-13	12	
		5	6					
		3	6	9999	12	0	6	S
		2	6			-1	6	S
	4	0	6	9999	12	0	12	S
		-1	6					S
		-3	6	-2	12	-5	12	S
		-5	6					S
27	1	0	12	-4	12	-4	12	S
								S
		5	12	1	12	0	6	S
						-1	6	S
	2	3	12	0	12	0	12	S
								S
		3	12	-2	12	2	12	S
								S
	3	9999	12	-4	12	-5	12	S
								S
		-4	12	-5	12	0	6	S

						-1	6		S
	4	-5	12	-4	12	0	12		S
									S
		-7	12	-2	12	2	12		S
									S
28	1	-5	12	-8	12	-7	6		S
						-5	6		S
		-7	6	-7	12	-4	24		S
		-9	6						S
	2	-7	12	-5	12				S
									S
		-10	12	-8	12	-5	6		S
						-7	6		S
	3	-4	12	-8	12	-9	24		S
		-5	12	-7	12				
	4	9999	12	9999	12	9999	12		
		-3	12	-7	12	3	12		
29	1	-1	12	-2	6	2	12		
				-5	6				
		0	12	-4	6	0	12		
				-7	18				
	2	-7	6			7	12		
		-9	6						
		-10	6	-8	12	-5	12		
		-12	6						
	3	-12	12	-7	24	0	240		
		0	6					S	
		-1	6					S	
	4	0	12	9999	12			S	
								S	
		-5	12	-3	12			S	
								S	
30	1	-4	12	-2	24			S	
								S	
		0	6					S	
		-1	6					S	
	2	0	12	9999	12			S	
								S	
		2	12	-2	12			S	
								S	
	3	-5	12	-2	12			S	
								S	
		0	6	-4	6			S	
		-1	6	-5	6			S	
	4	0	12	-4	12			S	
								S	
		2	12	-2	12			S	
								S	
31	1	-7	6	-5	12			S	

	-5	6				S
	-4	24	9999	12		S
						S
2			-8	12		S
						S
	-5	6	9999	12		S
	-7	6				S
3	-9	48	-7	48	7	S

Appendix F: Output of the Expert System for Fugue #2 in C minor (WTC I)

BAR	BEAT	MESSAGE	
		MINOR key	
		16TH	Base tempo: Quarter Note = 80. Accents: Beats 1 and 3
3	1	Voice 1:	Answer Statement
7	1	Voice 1:	Countersubject Statement
11	1	Voice 1:	Subject Statement
15	1	Voice 1:	Countersubject Statement
20	1	Voice 1:	Subject Statement
29	3	Voice 1:	Partial Subject/Answer Statement
1	1	Voice 2:	Subject Statement
3	1	Voice 2:	Countersubject Statement
15	1	Voice 2:	Answer Statement
20	1	Voice 2:	Countersubject Statement
7	1	Voice 3:	Subject Statement
11	1	Voice 3:	Countersubject Statement
26	3	Voice 3:	Subject Statement
		OPENING:	Slur the 2 16ths together.
			RULE22
1	2	SUBJECT:	Play both 8ths staccato.
			RULE16
1	3	SUBJECT:	Prolong the downbeat note slightly.
			RULE6
1	3	SUBJECT:	Slur the 2 16ths together.
			RULE22
1	4	SUBJECT:	Play both 8ths staccato.
			RULE16
2	1	SUBJECT:	Prolong the downbeat note slightly.
			RULE6
2	1	SUBJECT:	Slur the 2 16ths together.
			RULE22
2	2	SUBJECT:	Play both 8ths staccato.
			RULE16
2	3	SUBJECT:	Slur the 2 16ths together.
			RULE22
2	3	SUBJECT:	Slur the 16ths after the long note.
			RULE13
5	1	VOICE 1	Slur the 2 16ths together.
			RULE22
5	2	VOICE 1	Play both 8ths staccato.
			RULE16
5	3	VOICE 1	Prolong the downbeat note slightly.
			RULE6
5	3	VOICE 1	Slur the 2 16ths together.
			RULE22
5	4	VOICE 1	Play both 8ths staccato.
			RULE16
6	1	VOICE 1	Prolong the downbeat note slightly.
			RULE6
6	1	VOICE 1	Slur the 2 16ths together.

			RULE22
6	2	VOICE 1	Play both 8ths staccato.
			RULE16
6	3	VOICE 1	Prolong the downbeat note slightly.
			RULE6
6	3	VOICE 1	Slur the 2 16ths together.
			RULE22
6	4	VOICE 1	Slur the group of 16ths after long note.
			RULE26
7	3	VOICE 1	Prolong the downbeat note slightly.
			RULE6
7	4	VOICE 1	Play the last 8th of the beat staccato.
			RULE7
8	1	VOICE 1	Prolong the downbeat note slightly.
			RULE6
8	2	VOICE 1	Play the last 8th of the beat staccato.
			RULE7
8	3	VOICE 1	Prolong the downbeat note slightly.
			RULE6
8	4	VOICE 1	Play the last 8th of the beat staccato.
			RULE7
9	1	VOICE 1	Prolong the downbeat note slightly.
			RULE6
9	1	VOICE 1	Slur the 2 16ths together.
			RULE22
9	2	VOICE 1	Play both 8ths staccato.
			RULE16
9	4	VOICE 1	Play the last 8th of the beat staccato.
			RULE7
10	1	VOICE 1	Prolong the downbeat note slightly.
			RULE6
10	1	VOICE 1	Slur the 2 16ths together.
			RULE22
10	2	VOICE 1	Play both 8ths staccato.
			RULE16
10	4	VOICE 1	Play the last 8th of the beat staccato.
			RULE7
11	1	VOICE 1	Prolong the downbeat note slightly.
			RULE6
13	1	VOICE 1	Play 1st 16th stacc. & slur last 7 16ths
			RULE19
13	3	VOICE 1	Slur the group of 8 notes together.
			RULE1
14	1	VOICE 1	Play 1st 16th stacc. & slur last 7 16ths
			RULE19
14	3	VOICE 1	Slur the group of 8 notes together.
			RULE1
15	1	VOICE 1	Prolong the downbeat note slightly.
			RULE6
15	1	VOICE 1	Slur the 16ths together.
			RULE21
15	3	VOICE 1	Prolong the downbeat note slightly.
			RULE6
15	4	VOICE 1	Play the last 8th of the beat staccato.

			RULE7
16	1	VOICE 1	Prolong the downbeat note slightly.
			RULE6
16	2	VOICE 1	Play the last 8th of the beat staccato.
			RULE7
16	3	VOICE 1	Prolong the downbeat note slightly.
			RULE6
16	4	VOICE 1	Play the last 8th of the beat staccato.
			RULE7
17	1	VOICE 1	Prolong the downbeat note slightly.
			RULE6
17	1	VOICE 1	Slur the 2 16ths together.
			RULE22
17	3	VOICE 1	Slur the 2 16ths together.
			RULE22
18	1	VOICE 1	Slur the 2 16ths together.
			RULE22
18	3	VOICE 1	Slur the 2 16ths together.
			RULE22
19	1	VOICE 1	Slur the 2 16ths together.
			RULE22
19	3	VOICE 1	Slur the 2 16ths together.
			RULE22
22	1	VOICE 1	Slur the 2 16ths together.
			RULE22
22	2	VOICE 1	Play both 8ths staccato.
			RULE16
22	4	VOICE 1	Play the last 8th of the beat staccato.
			RULE7
23	1	VOICE 1	Prolong the downbeat note slightly.
			RULE6
23	1	VOICE 1	Slur the 2 16ths together.
			RULE22
23	2	VOICE 1	Play both 8ths staccato.
			RULE16
23	4	VOICE 1	Slur the group of 16ths after long note.
			RULE26
25	1	VOICE 1	Slur the 2 16ths together.
			RULE22
25	2	VOICE 1	Play both 8ths staccato.
			RULE16
25	3	VOICE 1	Prolong the downbeat note slightly.
			RULE6
25	3	VOICE 1	Slur the 2 16ths together.
			RULE22
25	4	VOICE 1	Play both 8ths staccato.
			RULE16
26	1	VOICE 1	Prolong the downbeat note slightly.
			RULE6
26	1	VOICE 1	Slur the 2 16ths together.
			RULE22
26	2	VOICE 1	Play both 8ths staccato.
			RULE16
26	3	VOICE 1	Play 1st 16th stacc. & slur last 7 16ths

			RULE19
27	1	VOICE 1	Prolong the downbeat note slightly.
			RULE6
27	2	VOICE 1	Play the last 8th of the beat staccato.
			RULE7
27	4	VOICE 1	Play the last 8th of the beat staccato.
			RULE7
28	1	VOICE 1	Prolong the downbeat note slightly.
			RULE6
28	1	VOICE 1	Slur the 2 16ths together.
			RULE22
28	2	VOICE 1	Play both 8ths staccato.
			RULE16
28	3	VOICE 1	Prolong the downbeat note slightly.
			RULE6
28	4	VOICE 1	Play the last 8th of the beat staccato.
			RULE7
29	1	VOICE 1	Prolong the downbeat note slightly.
			RULE6
29	2	VOICE 1	Slur the 4 16ths.
			RULE12
29	3	VOICE 1	Prolong the downbeat note slightly.
			RULE6
29	3	VOICE 1	Slur the 2 16ths together.
			RULE22
29	4	VOICE 1	Play both 8ths staccato.
			RULE16
30	1	VOICE 1	Prolong the downbeat note slightly.
			RULE6
30	1	VOICE 1	Slur the 2 16ths together.
			RULE22
30	2	VOICE 1	Play both 8ths staccato.
			RULE16
30	3	VOICE 1	Prolong the downbeat note slightly.
			RULE6
30	3	VOICE 1	Slur the 2 16ths together.
			RULE22
30	4	VOICE 1	Play both 8ths staccato.
			RULE16
31	1	VOICE 1	Slur the 2 16ths together.
			RULE22
31	2	VOICE 1	Slur the 2 16ths together.
			RULE22
3	1	VOICE 2	Slur the 16ths together.
			RULE21
3	3	VOICE 2	Prolong the downbeat note slightly.
			RULE6
3	4	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
4	1	VOICE 2	Prolong the downbeat note slightly.
			RULE6
4	2	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
4	3	VOICE 2	Prolong the downbeat note slightly.

			RULE6
4	4	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
5	2	VOICE 2	Slur the 16ths together.
			RULE21
5	3	VOICE 2	Slur the group of 16ths after long note.
			RULE26
6	1	VOICE 2	Slur the 16ths after the long note.
			RULE20
6	4	VOICE 2	Slur the 2 16ths together.
			RULE22
7	4	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
8	2	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
8	3	VOICE 2	Prolong the downbeat note slightly.
			RULE6
8	3	VOICE 2	Slur the 2 16ths together.
			RULE22
8	4	VOICE 2	Play both 8ths staccato.
			RULE16
9	2	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
9	3	VOICE 2	Prolong the downbeat note slightly.
			RULE6
9	3	VOICE 2	Slur the 2 16ths together.
			RULE22
9	4	VOICE 2	Play both 8ths staccato.
			RULE16
10	2	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
10	3	VOICE 2	Prolong the downbeat note slightly.
			RULE6
10	3	VOICE 2	Slur the 2 16ths together.
			RULE22
10	4	VOICE 2	Play both 8ths staccato.
			RULE16
11	2	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
11	3	VOICE 2	Prolong the downbeat note slightly.
			RULE6
11	4	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
12	2	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
12	3	VOICE 2	Slur the 2 16ths together.
			RULE22
12	4	VOICE 2	Play both 8ths staccato.
			RULE16
13	1	VOICE 2	Prolong the downbeat note slightly.
			RULE6
13	2	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
13	3	VOICE 2	Prolong the downbeat note slightly.

			RULE6
13	4	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
14	1	VOICE 2	Prolong the downbeat note slightly.
			RULE6
14	2	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
14	3	VOICE 2	Prolong the downbeat note slightly.
			RULE6
14	4	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
15	1	VOICE 2	Prolong the downbeat note slightly.
			RULE6
17	2	VOICE 2	Slur the 16ths together.
			RULE21
17	3	VOICE 2	Slur the group of 16ths after long note.
			RULE26
18	1	VOICE 2	Slur the 16ths after the long note.
			RULE13
18	3	VOICE 2	Prolong the downbeat note slightly.
			RULE6
18	3	VOICE 2	Slur the 2 16ths together.
			RULE22
18	4	VOICE 2	Play both 8ths staccato.
			RULE16
19	1	VOICE 2	Prolong the downbeat note slightly.
			RULE6
19	1	VOICE 2	Slur the 2 16ths together.
			RULE22
19	2	VOICE 2	Play both 8ths staccato.
			RULE16
19	3	VOICE 2	Prolong the downbeat note slightly.
			RULE6
19	3	VOICE 2	Slur the 2 16ths together.
			RULE22
19	4	VOICE 2	Play both 8ths staccato.
			RULE16
20	1	VOICE 2	Play 1st 16th stacc. & slur last 7 16ths
			RULE19
20	3	VOICE 2	Prolong the downbeat note slightly.
			RULE6
20	4	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
21	1	VOICE 2	Prolong the downbeat note slightly.
			RULE6
21	2	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
21	3	VOICE 2	Prolong the downbeat note slightly.
			RULE6
21	4	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
22	2	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
22	3	VOICE 2	Prolong the downbeat note slightly.

			RULE6
22	3	VOICE 2	Slur the 2 16ths together.
			RULE22
22	4	VOICE 2	Play both 8ths staccato.
			RULE16
23	2	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
23	3	VOICE 2	Prolong the downbeat note slightly.
			RULE6
23	3	VOICE 2	Slur the 2 16ths together.
			RULE22
23	4	VOICE 2	Play both 8ths staccato.
			RULE16
25	1	VOICE 2	Prolong the downbeat note slightly.
			RULE6
25	2	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
25	3	VOICE 2	Prolong the downbeat note slightly.
			RULE6
25	4	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
26	1	VOICE 2	Prolong the downbeat note slightly.
			RULE6
26	4	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
27	1	VOICE 2	Prolong the downbeat note slightly.
			RULE6
27	2	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
27	3	VOICE 2	Prolong the downbeat note slightly.
			RULE6
27	4	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
28	1	VOICE 2	Prolong the downbeat note slightly.
			RULE6
28	2	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
28	3	VOICE 2	Prolong the downbeat note slightly.
			RULE6
28	4	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
29	1	VOICE 2	Slur the 16ths together.
			RULE21
29	2	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
29	4	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
30	2	VOICE 2	Play the last 8th of the beat staccato.
			RULE7
30	3	VOICE 2	Prolong the downbeat note slightly.
			RULE6
30	3	VOICE 2	Slur the 2 16ths together.
			RULE22
30	4	VOICE 2	Play both 8ths staccato.

			RULE16
31	1	VOICE 2	Prolong the downbeat note slightly.
			RULE6
9	1	VOICE 3	Play 1st 16th stacc. & slur last 7 16ths
			RULE19
9	3	VOICE 3	Slur the group of 8 notes together.
			RULE1
10	1	VOICE 3	Play 1st 16th stacc. & slur last 7 16ths
			RULE19
10	3	VOICE 3	Slur the group of 8 notes together.
			RULE1
11	1	VOICE 3	Play 1st 16th stacc. & slur last 7 16ths
			RULE19
11	3	VOICE 3	Prolong the downbeat note slightly.
			RULE6
11	4	VOICE 3	Play the last 8th of the beat staccato.
			RULE7
12	1	VOICE 3	Prolong the downbeat note slightly.
			RULE6
12	2	VOICE 3	Play the last 8th of the beat staccato.
			RULE7
12	3	VOICE 3	Prolong the downbeat note slightly.
			RULE6
12	4	VOICE 3	Play the last 8th of the beat staccato.
			RULE7
13	1	VOICE 3	Prolong the downbeat note slightly.
			RULE6
13	2	VOICE 3	Play the last 8th of the beat staccato.
			RULE7
13	3	VOICE 3	Prolong the downbeat note slightly.
			RULE6
13	4	VOICE 3	Play the last 8th of the beat staccato.
			RULE7
14	1	VOICE 3	Prolong the downbeat note slightly.
			RULE6
14	2	VOICE 3	Play the last 8th of the beat staccato.
			RULE7
14	3	VOICE 3	Prolong the downbeat note slightly.
			RULE6
14	4	VOICE 3	Play the last 8th of the beat staccato.
			RULE7
15	1	VOICE 3	Prolong the downbeat note slightly.
			RULE6
15	4	VOICE 3	Play the last 8th of the beat staccato.
			RULE7
16	2	VOICE 3	Play the last 8th of the beat staccato.
			RULE7
16	3	VOICE 3	Prolong the downbeat note slightly.
			RULE6
16	3	VOICE 3	Slur the 2 16ths together.
			RULE22
16	4	VOICE 3	Play both 8ths staccato.
			RULE16
17	1	VOICE 3	Prolong the downbeat note slightly.

			RULE6
17	1	VOICE 3	Slur the 2 16ths together.
			RULE22
17	2	VOICE 3	Play both 8ths staccato.
			RULE16
17	3	VOICE 3	Prolong the downbeat note slightly.
			RULE6
17	3	VOICE 3	Slur the 2 16ths together.
			RULE22
17	4	VOICE 3	Play both 8ths staccato.
			RULE16
18	1	VOICE 3	Prolong the downbeat note slightly.
			RULE6
18	1	VOICE 3	Slur the 2 16ths together.
			RULE22
18	2	VOICE 3	Play both 8ths staccato.
			RULE16
18	4	VOICE 3	Slur the 16ths together.
			RULE21
19	1	VOICE 3	Slur the group of 16ths after long note.
			RULE26
19	3	VOICE 3	Slur the 16ths after the long note.
			RULE13
20	1	VOICE 3	Prolong the downbeat note slightly.
			RULE6
20	2	VOICE 3	Play the last 8th of the beat staccato.
			RULE7
20	3	VOICE 3	Prolong the downbeat note slightly.
			RULE6
20	4	VOICE 3	Play the last 8th of the beat staccato.
			RULE7
21	2	VOICE 3	Play the last 8th of the beat staccato.
			RULE7
21	3	VOICE 3	Prolong the downbeat note slightly.
			RULE6
21	3	VOICE 3	Slur the 2 16ths together.
			RULE22
21	4	VOICE 3	Play both 8ths staccato.
			RULE16
22	1	VOICE 3	Slur the group of 8 notes together.
			RULE1
22	3	VOICE 3	Play 1st 16th stacc. & slur last 7 16ths
			RULE19
23	1	VOICE 3	Slur the group of 8 notes together.
			RULE1
23	3	VOICE 3	Play 1st 16th stacc. & slur last 7 16ths
			RULE19
24	1	VOICE 3	Slur the group of 8 notes together.
			RULE1
24	3	VOICE 3	Play 1st 16th stacc. & slur last 7 16ths
			RULE19
25	3	VOICE 3	Slur the 16ths following the rest.
			RULE24
26	1	VOICE 3	Slur the 16ths together.

			RULE21
26	3	VOICE 3	Prolong the downbeat note slightly.
			RULE6
28	4	VOICE 3	Play the last 8th of the beat staccato.
			RULE7
29	1	VOICE 3	Prolong the downbeat note slightly.
			RULE6
29	2	VOICE 3	Play the last 8th of the beat staccato.
			RULE7

Appendix G: Source Code Listings

```
*************************************************************
************************* steps.prg
*****************************
*************************************************************
*** steps.prg
*** margaret johnson
*** dissertation project
*** written in clipper (a dbase compiler)

*** this program takes as input a numeric representation of
*** a fugue in 3 voices (in common time) and determines the
*** steps between successive notes. this data is put in the
*** STEPS field on the fugue database.
*************************************************************

* get file name from dos call
parameter filename
use &filename
go top

* start up the loop on voice 1
cnt = "1"
do while val(cnt) < 4
  do while .t.
   if note&cnt = "    ".or. note&cnt = "9999"
     skip
   else
     * store value of first note
     store val(note&cnt) to n&cnt
     skip
     * find the next note
     do while note&cnt = "    ".or. note&cnt = "9999"
       skip
       if eof()
         exit
       endif
     enddo
     if eof()
       exit
     endif
     * replace the STEP field with their difference
     replace step&cnt with ltrim(trim(str((val(note&cnt)- ;
        n&cnt))))
   endif
  enddo
  go top
  * next voice
  cnt = ltrim(trim(str(val(cnt)+1)))
enddo

*************************************************************
* end of steps.prg
```

```
**************************************************************
**********************************************************
******************** analyze.prg **********************
**********************************************************

*** analyze.prg
*** margaret johnson
*** dissertation project
*** written in clipper (a dbase compiler)

*** this program takes as input a numeric representation of
*** a fugue in 3 voices (in common time) and determines the
*** following things:

***              major or minor key
***              tempo (metronome mark)
***              subject
***              answer
***              main countersubject

*** the numeric representation is stored in a dbase
*** database, but we put it in an array which makes it
*** easier to work with in the program.

**************************************************************

* get file name & meter from dos call
parameter filename,meter_in
store ltrim(trim(filename)) to filename
store ltrim(trim(meter_in)) to meter_in

clear

* open the representation file & determine how long the
* fugue is ...
@ 0,1 say "Opening database and storing data to arrays."
select 1
use &filename
store reccount() to number_recs
* open other files
go top
select 2
use tempo
select 3
use results
zap

select 1

* first determine if the key is major or minor.  we do this
* by checking for minor thirds vs major thirds (3 half steps
* vs. 4).  if a melody  has more minor thirds than major
* thirds, then it is in a minor key.
```

```
count for val(note1)=3 to min_1
count for val(note2)=3 to min_2
count for val(note3)=3 to min_3
count for val(note1)=4 to maj_1
count for val(note2)=4 to maj_2
count for val(note3)=4 to maj_3
minor = min_1+min_2+min_3
major = maj_1+maj_2+maj_3

* generate output record for key
select 3
append blank
if minor > major
  key = "MINOR"
  replace message with "MINOR key"
else
  key = "MAJOR"
  replace message with "MAJOR key"
endif

@ 0,70 say key

select 1

* now do a lookup in the tempo table.  this table holds the
* base tempo based on meter and fastest note value.

* first, find what the fastest note value is.
go top
fastest = ""

* 32nd note first
count for val(rnote1) = 3 to r1
count for val(rnote2) = 3 to r2
count for val(rnote3) = 3 to r3

if (r1+r2+r3) = 0
        * check 16th notes
        count for val(rnote1) = 6 to r1
        count for val(rnote2) = 6 to r2
        count for val(rnote3) = 6 to r3
        if (r1+r2+r3) = 0
                * check 8th notes
                count for val(rnote1) = 12 to r1
                count for val(rnote2) = 12 to r2
                count for val(rnote3) = 12 to r3
                if (r1+r2+r3) = 0
                        * check quarter notes
                        count for val(rnote1) = 24 to r1
                        count for val(rnote2) = 24 to r2
                        count for val(rnote3) = 24 to r3
                        if (r1+r2+r3) = 0
                           * last one: half notes
```

```
                        fastest = "HALF"
                    else
                        fastest = "QUARTER"
                    endif
                else
                    fastest = "8TH"
                endif
            else
                fastest = "16TH"
            endif
else
            fastest = "32ND"
endif

select 2
locate for ltrim(trim(meter)) = meter_in .and.
trim(ltrim(fast_note)) = fastest
if .not. found()
   store fastest+"   No base tempo can be found." to say_it
else
   store fastest+"   Base tempo: "+ltrim(trim(tempo))+".;
         Accents: "+accents to say_it
endif

* generate output
select 3
append blank
replace message with say_it
@ 1,1 say say_it

select 1
go top

* initialize the arrays which will hold the numeric
* representation.  we initialize them to the actual size
* required by the fugue

declare beats[number_recs]
declare ;
voice1[number_recs],voiceR1[number_recs],voiceD1[number_recs]
declare ;
voice2[number_recs],voiceR2[number_recs],voiceD2[number_recs]
declare ;
voice3[number_recs],voiceR3[number_recs],voiceD3[number_recs]
declare ;
steps1[number_recs],steps2[number_recs],steps3[number_recs]

* load the data into the arrays
for i = 1 to number_recs
  beats[i] = BEAT
  voice1[i] = NOTE1
  voice2[i] = NOTE2
  voice3[i] = NOTE3
  voiceR1[i] = RNOTE1
```

```
   voiceR2[i] = RNOTE2
   voiceR3[i] = RNOTE3
   voiceD1[i] = iif(substr(STEP1,1,1) = "-","-","+")
   voiceD2[i] = iif(substr(STEP2,1,1) = "-","-","+")
   voiceD3[i] = iif(substr(STEP3,1,1) = "-","-","+")
   steps1[i] = STEP1
   steps2[i] = STEP2
   steps3[i] = STEP3
   skip
next

* one voice starts by stating the subject; we need to
* determine which voice this is.  we cannot just look at the
* first note, because the subject may start with a rest.  we
* need to look until we find something other than 9999 (rest)
* in one of the voices.  we set a stopping point of 20;
* surely something will have happened by then.  flag will
* hold which voice is the starter, and i will equal
* the number of notes from the beginning that the subject
* begins.  we exit as soon as we find one that is not a rest
* or a blank.

* note that sv stands for start_voice (in order to use the &
* operator in clipper, we need to keep the variable names
* short).

@ 2,1 say "Figuring out which voice is the subject & where;
        it starts."
sv = "0"
for i = 1 to 20

   if (voice1[i] != "9999").and.(voice1[i] != "    ")
           sv = "1"
           exit
   endif
   if (voice2[i] != "9999").and.(voice2[i] != "    ")
           sv = "2"
           exit
   endif
   if (voice3[i] != "9999").and.(voice3[i] != "    ")
           sv = "3"
           exit
   endif

next

* little error checking
if sv = "0"
   @ 24,1 say "Could not find the subject."
   inkey(0)
   return
endif

start_note = i
```

```
* next we need to find where the subject ends so the steps
* pattern can be stored in a char variable.  the subject ends
* roughly where the answer begins, so we need to look at the
* other two voices and find where they are not 9999 (rest) or
* blanks.  the first task is to determine what the other 2
* voices are.

@ 3,1 say "Finding the answer."

do case
case sv = "1"
  o1 = "2"
  o2 = "3"
case sv = "2"
  o1 = "1"
  o2 = "3"
case sv = "3"
  o1 = "1"
  o2 = "2"
endcase

* now find the answer voice and where it starts. note, av
* stands for answer voice.

av = "0"
for i = 1 to 50

  if (voice&o1[i] != "9999").and.(voice&o1[i] != "    ")
          av = o1
          exit
  endif
  if (voice&o2[i] != "9999").and.(voice&o2[i] != "    ")
          av = o2
          exit
  endif

next

answ_strt = i

* the subject rarely ends right where the answer begins. the
* following rules apply in common time fugues:

* if the answer starts on a beat other than 1 or 3:
*       if answer starts on downbeat 2 or 4,
*          then the subject ends on current downbeat - 1
*       if answer does not start on a downbeat,
*          then the subject ends on the downbeat of the
*          current beat
*
* if the answer starts on beat 1 or 3:
*    then the subject ends on the same beat
```

```
if beats[answ_strt] = "1".or.beats[answ_strt] = "3"
   subj_end = answ_strt
   right_on = "T"
else
   right_on = "F"
   if beats[answ_strt] = "2".or.beats[answ_strt] = "4"
      for i = (answ_strt-1) to 1 step -1
        if beats[i] != "  "
          subj_end = i
          exit
        endif
      next
        else
      for i = answ_strt to 1 step -1
        if beats[i] != "  "
          subj_end = i
          exit
        endif
      next
   endif
endif

* now, store the step patterns of the subject and answer in
* character variables since we will have to do a lot with
* them.  both of these melodies will have the same length.

@ 4,1 say "Analyzing subject and answer."

s_eq_a = "F"
subject = ""
subject_M = ""
subject_R = ""
subject_D = ""
answer = ""
answer_M = ""
answer_R = ""

answ_end = subj_end

for i = 1 to subj_end
   subject = subject + steps&sv[i]
   subject_M = subject_M + voice&sv[i]
   subject_R = subject_R + voiceR&sv[i]
   subject_D = subject_D + iif(substr(steps&sv[i],1,1) =;
         "-","-","+")
next

* strip off any leading rests; and the first step because
* this will be based on what came prior to the first note of
* the subject, which is not relevant

do while substr(subject_M,1,4) = "    ".or.;
        substr(subject_M,1,4) = "9999"
```

```
   subject = stuff(subject,1,4,"")
   subject_R = stuff(subject_R,1,4,"")
   subject_M = stuff(subject_M,1,4,"")
   subject_D = stuff(subject_D,1,1,"")
   subj_end = subj_end - 1
enddo

* and the first note off
subject = substr(subject,5,(subj_end*4)-1)
subject_R = substr(subject_R,5,(subj_end*4)-1)
subject_D = substr(subject_D,2,subj_end)

* now for the answer
acnter = 0

for i = answ_strt to (answ_end * 4)
    answer = answer + steps&av[i]
    answer_M = answer_M + voice&av[i]
    answer_R = answer_R + voiceR&av[i]
    acnter = acnter + 1
    if acnter > (subj_end-1)
      exit
    endif
next

* and the first note off
answer = substr(answer,5,(answ_end*4)-1)

* if subject = answer, we will call them all subject
if subject = answer
  s_eq_a = "T"
endif

* next come the countersubjects.  these are very tricky
* because they may or may not play against the entire length
* of the subject or answer; also, the fugue may not have any
* countersubjects -  sometimes the subject or answer is
* accompanied by new material each time.  the first statement
* of the first countersubject is the melody played by the
* subject voice right after it finishes with the subject;
* this melody accompanies the answer.  thus, the start of CSI
* coincides with the start of the answer (answ_strt).
* the problem is figuring out where it ends.  another set of
* rules apply, similar to ones for subject end.

@ 5,1 say "Searching for a main countersubject."

* if the answer starts on a beat other than 1 or 3:
*       if answer starts on downbeat 2 or 4,
*          then CSI begins on the downbeat
*       else if answer does not start on a downbeat,
*          if next downbeat is 2 or 4
*             then CSI begins on the current beat
```

```
*           else next downbeat is 1 or 3
*              then CSI begins on the 1 or 3 downbeat

* else if answer starts on 1 or 3
*      then CSI begins on the downbeat

if beats[answ_strt] = "1".or.beats[answ_strt] = "3"
   cs_start = answ_strt
else
   if beats[answ_strt] = "2".or.beats[answ_strt] = "4"
      cs_start = answ_strt
   else
      for i = answ_strt to answ_strt * 2
        do case
        case beats[i] = "2 ".or. beats[i] = "4 "
          cs_start = answ_strt
          exit
        case beats[i] = "1 ".or. beats[i] = "3 "
          cs_start = i
          exit
        otherwise
          loop
        endcase
      next
   endif
endif

* now to find the end of it, we need to search for the
* rhythmic pattern of the countersubject elsewhere in the
* piece. we do rhythm here because there is a bit more
* freedom in the composition of CS's than in subject and
* answer.  if the CS has the general sound of the original
* CS, it should be called a CS even though the melodic
* pattern may not be exactly the same. we may not find the
* pattern at all, which means there are no CS. we will use an
* the at() function to find the pattern in the array. to use
* at(), we need to put the arrays into string variables.  we
* will look for it in the other voices  since we know we will
* find it in the subject voice.

textR = ""
for i = 1 to number_recs
  textR = textR + voiceR&o1[i]
next
textD = ""
for i = 1 to number_recs
  textD = textD + voiceD&o1[i]
next

* shorten will be used to take notes off the end of the CS
* until we find a match
shorten = 0
```

```
do while shorten < subj_end

  cs_R = ""
  cs_D = ""
  for i = cs_start to ((cs_start+acnter+1) - shorten)
     cs_R = cs_R + voiceR&sv[i]
     cs_D = cs_D + voiceD&sv[i]
  next

  where = 1
  do while where > 0
    where = at(cs_R,textR)
    if where > 0
      if substr(textD,round(int(where)/4,0)+1,len(cs_D)) =;
          cs_D
        * have to divide by 4 to get the actual value since
        * the array records have a length of 4
        where = round(int(where)/4,0) + 1
        exit
      else
        * look for another one
        textR = stuff(textR,where,4,space(4))
      endif
    else
     shorten = shorten + 1
     exit
    endif
  enddo
  if where > 0
    exit
  endif

enddo

* if we have not found it yet, look in the other voice

if where = 0

  textR = ""
  for i = 1 to number_recs
    textR = textR + voiceR&o2[i]
  next
  textD = ""
  for i = 1 to number_recs
    textD = textD + voiceD&o2[i]
  next

 do while shorten < subj_end

  cs_R = ""
  cs_D = ""
  for i = cs_start to ((cs_start+acnter+1) - shorten)
     cs_R = cs_R + voiceR&sv[i]
```

```
      cs_D = cs_D + voiceD&sv[i]
    next

    where = 1
    do while where > 0
      where = at(cs_R,textR)
      if where > 0
        if substr(textD,round(int(where)/4,0)+1,len(cs_D)) =;
                cs_D
          * have to divide by 4 to get the actual value since
          * the array records have a length of 4
          where = round(int(where)/4,0) + 1
          exit
        else
          * look for another one
          textR = stuff(textR,where,4,space(4))
        endif
      else
       shorten = shorten + 1
       exit
      endif
    enddo
    if where > 0
      exit
    endif

  enddo

endif

* if where is still 0, there are no CS
if where = 0
  @ 6,1 say "No countersubject found."
  cs_R = "NONE"
else
  @ 6,1 say "Countersubject found."
  cs_length = acnter - shorten
  * strip off last note of cs_R which is not really a part of
* the pattern if it starts on beat 1 or 3
  if right_on = "T"
    cs_R = substr(cs_R,1,len(trim(cs_R))-4)
  endif
  * store the main countersubject melodic pattern too
endif

* now we need to label the melodies in the database.  we do
* this by finding all occurences of the 3 main melodies,
* accessing the database and putting "S" "A" or "CS" in the
* MELODY fields.

@ 7,1 say "Finding all subjects and marking them in;
           database."

vcnt = "1"
```

```
scnt1 = 0
scnt2 = 0
scnt3 = 0
scnt4 = 0

* subject first
do while val(vcnt) < 4

  * initialize some variables
  for i = 1 to 15
    m = ltrim(trim(str(i)))
    found_it&m = " "
  next

  text = ""
  for i = 1 to number_recs
    text = text + steps&vcnt[i]
  next

  cnt = "1"
  where = 1
  do while where > 0

    where = at(subject,text)
    if where > 0
      found_it&cnt = round(int(where)/4,0) + 1
      * blank out subject where it was found in text so we
        * can find next one
      text = stuff(text,where,subj_end,space(subj_end))
      cnt = ltrim(trim(str(val(cnt)+1)))
      scnt&vcnt = scnt&vcnt + 1
    endif
  enddo

  mm = "1"
  do while val(mm) < val(cnt)
    * replace the code in the database
    goto found_it&mm
    skip -1
    for i = 1 to subj_end
      replace melody&vcnt with "S    "
      skip
    next
    mm = ltrim(trim(str(val(mm)+1)))
  enddo

  vcnt = ltrim(trim(str(val(vcnt)+1)))

enddo


* now do the same for answers
@ 8,1 say "Finding all answers and marking them in database."
```

```
vcnt = "1"
acnt1 = 0
acnt2 = 0
acnt3 = 0
acnt4 = 0

if s_eq_a = "F"

  do while val(vcnt) < 4

    for i = 1 to 15
      m = ltrim(trim(str(i)))
      found_it&m = " "
    next

    text = ""
    for i = 1 to number_recs
      text = text + steps&vcnt[i]
    next

    cnt = "1"
    where = 1
    do while where > 0

      where = at(answer,text)
      if where > 0
        found_it&cnt = round(int(where)/4,0) + 1
        * blank out answer where it was found in text so we
        * can find next one
        text = stuff(text,where,subj_end,space(subj_end))
        cnt = ltrim(trim(str(val(cnt)+1)))
        acnt&vcnt = acnt&vcnt + 1
      endif

    enddo

    mm = "1"
    do while val(mm) < val(cnt)
      * replace the code in the database
      goto found_it&mm
      skip -1
      for i = 1 to subj_end
        replace melody&vcnt with "A    "
        skip
      next
      mm = ltrim(trim(str(val(mm)+1)))
    enddo

    vcnt = ltrim(trim(str(val(vcnt)+1)))

  enddo

endif
```

```
* and finally, countersubject - if there is one
if cs_R != "NONE"
  @ 9,1 say "Finding all main countersubjects and marking ;
             them in database."

  vcnt = "1"

  do while val(vcnt) < 4

    * initialize some variables
    for i = 1 to 15
      m = ltrim(trim(str(i)))
      found_it&m = " "
    next

    textR = ""
    for i = 1 to number_recs
      textR = textR + voiceR&vcnt[i]
    next
    textD = ""
    for i = 1 to number_recs
      textD = textD + voiceD&vcnt[i]
    next

    cnt = "1"
    where = 1
    do while where > 0

     where = at(cs_R,textR)
     if where > 0
      if substr(textD,round(int(where)/4,0)+1,len(cs_D)) =;
                cs_D
        found_it&cnt = round(int(where)/4,0) + 1
        * blank out CS where it was found in text so we can
        * find next one
        textR = stuff(textR,where,4,space(4))
        cnt = ltrim(trim(str(val(cnt)+1)))
      else
        textR = stuff(textR,where,4,space(4))
      endif
     endif
    enddo

    mm = "1"
    do while val(mm) < val(cnt)
      * replace the code in the database
      goto found_it&mm
      if right_on = "T"
        increm = 1
      else
        increm = 3
      endif
      for i = 1 to (cs_length+increm)
```

```
        replace melody&vcnt with "CS   "
        skip
      next
      mm = ltrim(trim(str(val(mm)+1)))
    enddo

    vcnt = ltrim(trim(str(val(vcnt)+1)))

  enddo

else
  @ 9,1 say "No countersubjects to mark."
endif

* finally we need to find partial subjects and answers.  we
* have one pre-condition for a partial: it must be at least 1
* bar long, otherwise it can really be considered a motive
* not an actual partial statement of the melody.  So we take
* the first bar of the subject (rhythm and direction of the
* melody), count how many times it appears in the voices; if
* it appears more than the total number of times the full
* subject or answer appears, then we have some partials.

* we mark these in the database as "PT".  we only mark their
* starting notes since it may vary how much more of the
* subject or answer follows the first bar.  all we really
* care about here is that it would sound as though the
* subject or answer is really being stated, but it does not
* end up the same each time. also, it may not be an exact
* statement, but rather have the general feel of it in rhythm
* and melodic direction.

@ 10,1 say "Looking for partial subjects/answers and marking;
            them in database."

vcnt = "1"
pscnt1 = 0
pscnt2 = 0
pscnt3 = 0
pscnt4 = 0

* create a character string of the first bar of the subject
* rhythm
psubj_r = substr(subject_R,1,64)
* create a character string of the first bar of the subject
* direction
psubj_d = substr(subject_D,1,16)

do while val(vcnt) < 4

  * initialize some variables
  for i = 1 to 15
    m = ltrim(trim(str(i)))
    found_it&m = " "
```

```
  next

  textR = ""
  for i = 1 to number_recs
    textR = textR + voiceR&vcnt[i]
  next
  textD = ""
  for i = 1 to number_recs
    textD = textD + voiceD&vcnt[i]
  next

  cnt = "1"
  where = 1
  do while where > 0

    where = at(psubj_r,textR)
    if where > 0
      if substr(textD,round(int(where)/4,0)+1,16) = psubj_d
        found_it&cnt = round(int(where)/4,0) + 1
        * blank out subject where it was found in text so we
        * can find next one
        textR = stuff(textR,where,16,space(16))
        cnt = ltrim(trim(str(val(cnt)+1)))
        pscnt&vcnt = pscnt&vcnt + 1
      else
        textR = stuff(textR,where,16,space(16))
      endif
    endif
  enddo

  if (scnt&vcnt+acnt&vcnt) < pscnt&vcnt
    mm = "1"
    do while val(mm) < val(cnt)
      * replace the code in the database
      goto found_it&mm
      skip -1
      if melody&vcnt ="S    ".or.melody&vcnt ="A    "
      else
         replace melody&vcnt with "PT   "
      endif
      mm = ltrim(trim(str(val(mm)+1)))
    enddo
  endif
  vcnt = ltrim(trim(str(val(vcnt)+1)))

enddo

* now we need to put some output in results.dbf.  we will put
* where the subject, answer, cs and partial melodies are
* located.

vcnt = "1"
```

```
do while val(vcnt) < 4

  select 1
  go top
  store "X" to old_one

  do while .not. eof()

     if (MELODY&vcnt != "    ").and.(MELODY&vcnt != old_one)
        store MELODY&vcnt to m_test
        store MELODY&vcnt to old_one
        store recno() to rec_mark
        select 3
        append blank
     else
        store MELODY&vcnt to old_one
        skip
        loop
     endif

     do case
     case m_test = "S"
           replace message with "Voice "+vcnt+":  Subject;
           Statement"
     case m_test = "A"
           replace message with "Voice "+vcnt+":  Answer;
           Statement"
     case m_test = "CS"
           replace message with "Voice "+vcnt+":
           Countersubject Statement"
     case m_test = "PT"
           replace message with "Voice "+vcnt+":  Partial;
           Subject/Answer Statement"
     endcase

     store int((rec_mark+16)/4)/4 to bar_mark
     store ltrim(trim(str(int(bar_mark)))) to bar_mark
     store rec_mark - ((val(bar_mark)-1)*16) to beat_flg
     do case
     case beat_flg >= 1 .and. beat_flg < 5
       beat_mark = "1"
     case beat_flg >= 5 .and. beat_flg < 9
       beat_mark = "2"
     case beat_flg >= 9 .and. beat_flg < 13
       beat_mark = "3"
     case beat_flg >= 13
       beat_mark = "4"
     endcase

     replace BAR with bar_mark, BEAT with beat_mark

     select 1
     do while (MELODY&vcnt != "    ").and.(MELODY&vcnt =;
        old_one)
```

```
        skip
        if eof()
          exit
        endif
      enddo

    enddo

  vcnt = ltrim(trim(str(val(vcnt)+1)))

enddo

*****************************************************************
*** end of analyze.prg
*****************************************************************




*****************************************************************
************************ process.prg ***************************
*****************************************************************

*** process.prg
*** margaret johnson
*** dissertation project
*** written in clipper (a dbase compiler)

*** this program takes as input a numeric representation of
*** a fugue in 3 voices (in common time) which has been
*** analyzed by analyze.prg. here is where we apply the
*** expertise to determine a suitable interpretation of the
*** subject.

*****************************************************************

* get file name from dos call
parameter filename
store ltrim(trim(filename)) to filename

clear
set exact off

* open the representation file & all the other required files
@ 0,1 say "Processing "+filename
select 1
use &filename
* FA for start of fugue
select 2
use open_FA
* FA for rest of fugue
select 3
use FA
```

```
* rules
select 4
use rule
* place for the output
select 5
use results

select 1
go top

* determine which voice starts: this voice will have an "S"
* in its melody line

do while .t.

   do case
   case MELODY1 = "S"
      vox = "1"
      exit
   case MELODY2 = "S"
      vox = "2"
      exit
   case MELODY3 = "S"
      vox = "3"
      exit
   otherwise
      skip
   endcase

enddo

*****************************************************************
*** first look at the opening of the fugue which is handled
*** differently than the rest; we have a special FA for the
*** opening
*****************************************************************

action = ""

* special case for rests as the opening figure; we need to
* look at the actual note not the rhythm
if NOTE&vox = "9999"
   note_1 = "99"
else
   store RNOTE&vox to note_1
endif

* find the opening note in open_FA start state
select 2
locate for ltrim(trim(NODE1)) = ltrim(trim(note_1))

if found()

        select 1
```

```
          start_mark = recno()
          current = start_mark
          cnt = "2"
          do while .t.
            * get the lookahead
            select 1
            skip
            do while RNOTE&vox = "    "
               skip
               current = current + 1
            enddo
            store RNOTE&vox to note_&cnt
            store NOTE&vox to mel_&cnt

            select 2
            * check if lookahead matches the pattern
            if ltrim(trim(NODE&cnt)) = ltrim(trim(note_&cnt));
                  .and. mel_&cnt != "9999"
               * it does, so continue on to check the next note
               select 1
               cnt = ltrim(trim(str(val(cnt)+1)))
               current = recno()
            * it does not match pattern, so we have reached
            * the end or  we have failed
            else
               * need action from previous node
               cnt = ltrim(trim(str(val(cnt)-1)))
               store ACT&cnt to action
               * fail
               if action = "    "
                  action = "FAIL IN MIDDLE ON OPENING SEARCH"
               endif
               exit
            endif
          enddo

else

   * we found nothing in the opening pattern
   action = "FAIL ALL ON OPENING SEARCH"

endif

select 1
* now process if we did find something
if action != "FAIL"
  * set up for next round
  store current+1 to start_mark
  * if action is RULE, just do a lookup in the rule dbf,
  * otherwise  have to run a PROC
  if action != "RULE"
    do &action with start_mark,current,vox,action
  endif
  * we have either returned from a PROC or have a RULE at
```

```
  * this point
  if action != "    "
     select 4
     * get the rule
     locate for ltrim(trim(NUMBER)) = ltrim(trim(action))
     store THEN to to_do
     * dump it in the output
     select 5
     append blank
     replace message with "OPENING:  "+to_do+action
     select 1
  endif
else
  store current to start_mark
endif

*************************************************************
*** now work on the rest of the subject
*************************************************************

search_again = "F"
no_skip_flag = "T"

do while MELODY&vox = "S"

     goto start_mark
     * no_skip_flag is used for a special case where the last
        * note of a subject or answer must be included in the
        * next pattern.  it is used in PROC5
     if no_skip_flag = "T"
        * there are blank records depending on the rhythmic
        * value.
        do while (RNOTE&vox = "    ".or. NOTE&vox =;
        "9999").and..not.eof()
           skip
        enddo
     else
        no_skip_flag = "T"
     endif

     * initialize for this round
     start_mark = recno()
     current = start_mark
     cnt = "1"
     action = ""
     * find the first note
     store ltrim(trim(RNOTE&vox)) to note_1
     store NOTE&vox to mel_&cnt

     * special check for rule 6 here which overrides the
        * rules in FA,  rule 6 is about prolonging a downbeat
        * note
     if (note_1 = "12").and.(BEAT="1 ".or.BEAT="3 ")
        action ="RULE6"
```

```
else

   select 3
   * search_again is used for determining if we should
   * start at the top of the FA database or continue
   * from where we already are.
   * this is used because note values 6, 12, 18 have
   * more than one  transition out
   if search_again = "F"
      locate for ltrim(trim(NODE1)) = note_1
   else
      search_again = "F"
      continue
   endif

   if .not. found()
      * reset our starting position and try again
      start_mark = start_mark + 1
      select 1
      loop
   else
      select 1
      cnt = ltrim(trim(str(val(cnt)+1)))
   endif

   do while .t.
      * get the lookahead
      skip
      do while RNOTE&vox = "    ".and.MELODY&vox = "S"
         skip
      enddo
      store RNOTE&vox to note_&cnt
      store NOTE&vox to mel_&cnt

      if MELODY&vox = "S"
        select 3
        * check if lookahead matches pattern in FA
        if ltrim(trim(NODE&cnt)) =;
           ltrim(trim(note_&cnt)) .and. mel_&cnt !=;
           "9999"
             * it does so try the next note
             select 1
             cnt = ltrim(trim(str(val(cnt)+1)))
             current = recno()
         else
             * we are at the end of a pattern or we have
             * failed;  need action from previous node
             * if it is  not empty
             cnt = ltrim(trim(str(val(cnt)-1)))
             store ACT&cnt to action
             * failed
             if action = "    "
                * 6/12/18 rhythmic values have more than
                * one sequence
```

```
                    * associated with them, so we want to
                    * try the next one rather than skip the
                    * whole thing
                    if note_1 = "6".or.note_1 =;
                    "12".or.note_1 = "18"
                       search_again = "T"
                    else
                       search_again = "F"
                       start_mark = start_mark + 1
                    endif
                 endif
                 exit
              endif
           else
                * process the last pattern of the subject
                select 3
                * need action from previous node if it's not
                * empty
                cnt = ltrim(trim(str(val(cnt)-1)))
                store ACT&cnt to action
                exit
           endif
        enddo
endif

* now figure out the rule/proc that applies
select 1
* the search failed
if action = "    "
   loop
else
   * run a PROC or process a RULE
   if action != "RULE"
     do &action with start_mark,current,vox,action
   endif
   * on return from proc5, we need to try the set of 4
   * 16ths
   if action = "TRY AGAIN ON 4"
       action = "    "
       current = start_mark + 3
       do PROC1 with start_mark,current,vox,action
   endif
   if action != "    "
      * figure out the bar and the beat
      * divide by 4 because 16th note is fastest note
      store int((start_mark+16)/4)/4 to bar_mark
      store ltrim(trim(str(int(bar_mark)))) to bar_mark
      store start_mark - ((val(bar_mark)-1)*16) to;
            beat_flg
      do case
      case beat_flg >= 1 .and. beat_flg < 5
         beat_mark = "1"
      case beat_flg >= 5 .and. beat_flg < 9
         beat_mark = "2"
```

```
            case beat_flg >= 9 .and. beat_flg < 13
               beat_mark = "3"
            case beat_flg >= 13
               beat_mark = "4"
            endcase

           * get the rule
           select 4
           locate for ltrim(trim(NUMBER)) = ;
              ltrim(trim(action))
           store THEN to to_do
           * dump it in the output
           select 5
           append blank
           replace message with "SUBJECT:  "+to_do+action
           replace records with start_mark
           replace bar with bar_mark
           replace beat with beat_mark
           select 1
         endif
         store current+1 to start_mark
      endif

enddo

*****************************************************************
*** finally, the rest of the piece
*****************************************************************

vox = "1"
vox_start = start_mark
no_skip_flag = "T"

do while vox != "4"

  * reset for processing a new voice
  select 1
  start_mark = vox_start
  goto start_mark
  search_again = "F"

  do while .not. eof()

     goto start_mark
     * no_skip_flag is used for a special case where the last
        * note of
     * a subject or answer must be included in the next
        * pattern.  it is used in PROC5
     if no_skip_flag = "T"
        * we do everything but the subject & answer since we
        * already did those
        do while (RNOTE&vox = "    ".or.MELODY&vox =;
        "S".or.MELODY&vox = "A";
        .or. NOTE&vox = "9999").and..not.eof()
```

```
      skip
   enddo
   if eof()
      exit
   endif
else
  no_skip_flag = "T"
endif

* get ready for the next round
start_mark = recno()
current = start_mark
cnt = "1"
action = ""

* find the first note
store ltrim(trim(RNOTE&vox)) to note_1

* special check for rule 6 here which overrides the
   * rules in FA
if (note_1 = "12").and.(BEAT="1 ".or.BEAT="3 ")
   action ="RULE6"
else

   select 3
   * search_again is used for determining if we should
   * start at the top of the FA database or continue
   * from where we already are.
   * this is used because note values 6, 12, 18 have
   * more than one transition out
   if search_again = "F"
      locate for ltrim(trim(NODE1)) = note_1
   else
      search_again = "F"
      continue
   endif

   if .not. found()
      * cannot match the first note, so reset and try
      * the next one
      start_mark = start_mark + 1
      select 1
      loop
   else
      * we found the first note
      select 1
      cnt = ltrim(trim(str(val(cnt)+1)))
   endif

   do while .t.
      * get the lookahead
      skip
      do while RNOTE&vox = "    ".and..not. eof()
         skip
```

```
        enddo
        if eof()
          action = "    "
          exit
        endif
        store RNOTE&vox to note_&cnt
        store NOTE&vox to mel_&cnt

        select 3
        * check if the lookahead matches our current
        * pattern in FA
        if ltrim(trim(NODE&cnt)) = ltrim(trim(note_&cnt));
             .and. mel_&cnt != "9999"
          * it does, so try the next note
          select 1
          cnt = ltrim(trim(str(val(cnt)+1)))
          current = recno()
        else
          * it doesn't, so we have either failed or
          * finished
          * need action from previous node if it's not
          * empty
           cnt = ltrim(trim(str(val(cnt)-1)))
           store ACT&cnt to action
           * we have failed
           if action = "    "
               * 6/12/18 rhythmic values have more than
               * one sequence associated with them, so we
               * want to try the next  one rather than
               * skip the whole thing
               if note_1 = "6".or.note_1 = "12".or.note_1;
                     = "18"
                   search_again = "T"
               else
                   search_again = "F"
                   start_mark = start_mark + 1
               endif
           endif
           exit
        endif

    enddo
endif

* process action
select 1
* failed
if action = "    "
    loop
else
    * we have a rule or proc that applies
    if action != "RULE"
       do &action with start_mark,current,vox,action
    endif
```

```
         * on return from proc5, we need to try the set of 4
         * 16ths
         if action = "TRY AGAIN ON 4"
            action = "    "
            current = start_mark + 3
            do PROC1 with start_mark,current,vox,action
         endif
         * process the rule
         if action != "    "
            * figure out the bar and the beat
            * divide by 4 because 16th note is fastest note
            store int((start_mark+16)/4)/4 to bar_mark
            store ltrim(trim(str(int(bar_mark)))) to bar_mark
            store start_mark - ((val(bar_mark)-1)*16) to;
                  beat_flg
            do case
            case beat_flg >= 0 .and. beat_flg < 5
              beat_mark = "1"
            case beat_flg >= 5 .and. beat_flg < 9
              beat_mark = "2"
            case beat_flg >= 9 .and. beat_flg < 13
              beat_mark = "3"
            case beat_flg >= 13
              beat_mark = "4"
            endcase

            select 4
            * find the rule
            locate for ltrim(trim(NUMBER)) =;
                  ltrim(trim(action))
            store THEN to to_do
            select 5
            * dump the output in the file
            append blank
            replace message with "VOICE "+vox+"  ;
                  "+to_do+action
            replace records with start_mark
            replace bar with bar_mark
            replace beat with beat_mark
            select 1
         endif
         store current+1 to start_mark
      endif

   enddo

   * reset for the next voice
   vox = ltrim(trim(str(val(vox)+1)))

enddo

*************************************************************
* end of main part of process.prg;  PROCS follow:
*************************************************************
```

```
**********************************************************************
procedure PROC1
parameter start_mark,current,vox,action

* this procedure determines which rule a group of 4 16ths
* applies to. the possible rules are 12, 14, 15; we need to
* determine if the 4 notes are stepwise, or if there is a
* leap between the 1st and 2nd or 3rd and 4th.  stepwise
* motion has successive STEPS fields of 1 or 2.

select 1
goto start_mark
skip
stepwise = "T"
leap1 = "F"
leap2 = "F"
action = "    "

* check rule 12 first: stepwise 16ths are slurred
cnting = 1
do while cnting <= 3
  if abs(val(STEP&vox)) = 1 .or. abs(val(STEP&vox)) = 2
    stepwise = "T"
  else
    stepwise = "F"
    exit
  endif
  skip
  do while STEP&vox = "    "
    skip
  enddo
  cnting = cnting + 1
enddo

if stepwise = "T"
   action = "RULE12"
   return
else
   * now check rule 14: leap between 1st and 2nd?
   goto start_mark
   skip
   if abs(val(STEP&vox)) = 1 .or. abs(val(STEP&vox)) = 2
     leap1 = "F"
     * then check rule 15: leap between 3rd and 4th?
     * first 3 are stepwise and then a leap
     skip
     do while STEP&vox = "    "
       skip
     enddo
        if abs(val(STEP&vox)) = 1 .or. abs(val(STEP&vox)) = 2
       leap2 = "T"
     else
       leap2 = "F"
     endif
```

```
      if leap2 = "T"
        skip
        do while STEP&vox = "    "
          skip
        enddo
        * we want a leap here
        if abs(val(STEP&vox)) = 1 .or. abs(val(STEP&vox)) = 2
          leap2 = "F"
        else
          leap2 = "T"
        endif
      endif
    else
      * check rule 15
      * there is a leap between 1 & 2
      * but we need to check that the rest is stepwise
      skip
      do while STEP&vox = "    "
         skip
      enddo
      if abs(val(STEP&vox)) = 1 .or. abs(val(STEP&vox)) = 2
         leap1 = "T"
      else
         leap1 = "F"
      endif
      if leap1 = "T"
         skip
         do while STEP&vox = "    "
            skip
         enddo
         if abs(val(STEP&vox)) = 1 .or. abs(val(STEP&vox)) = 2
            leap1 = "T"
         else
            leap1 = "F"
         endif
      endif
    endif
endif

if leap1 = "T"
  action = "RULE14"
endif
if leap2 = "T"
  action = "RULE15"
endif

goto current

*****************************************************************
* end of PROC1
*****************************************************************
```

```
*****************************************************************
procedure PROC2
parameter start_mark,current,vox,action

* this procedure determines which rule a rest-8th-8th-8th
* applies to. the possible rules are 3, 4, 5; we need to
* determine if the key is major or minor and if the 8ths are
* stepwise or leaping.  the STEPS field tells us about leaps
* vs. steps. the key is at the top of the results file

select 5
go top
store ltrim(trim(message)) to key_type

select 1
goto start_mark
stepwise = "T"

if key_type = "MINOR"
   action = "RULE5"
   return
endif

* check for stepwise/leaps
cnting = 1
do while cnting <= 3
    skip
    do while STEP&vox = "    "
      skip
    enddo
    if abs(val(STEP&vox)) = 1 .or. abs(val(STEP&vox)) = 2
      stepwise = "T"
    else
      stepwise = "F"
      exit
    endif
    cnting = cnting + 1
enddo

if stepwise = "T"
   action = "RULE3"
else
   action = "RULE4"
endif

goto current

*****************************************************************
* end of PROC2
*****************************************************************
```

```
*****************************************************************
procedure PROC3
parameter start_mark,current,vox,action

* this procedure determines which rule applies to a sequence
* of 2 8ths the possible rules are 7, 16, 17, 18; we need to
* determine if the 8ths are stepwise or leaping. the STEPS
* field tells us about leaps vs. steps.

select 1
goto start_mark

* first, check if the first eighth is not on a downbeat.
* this tells us if rule 7 might apply
if BEAT = "  "
  * find out if next beat is 1 or 3 since the accents are on
   * 1 or 3
  do while BEAT = "  "
    skip
  enddo
  if BEAT = "1" .or. BEAT = "3"
    action = "RULE7"
    current = current - 1
  else
    action = "     "
  endif
  return
endif

skip
do while STEP&vox = "    "
   skip
enddo

do case
case abs(val(STEP&vox)) = 0
  action = "RULE17"
case abs(val(STEP&vox)) = 1
  action = "RULE18"
otherwise
  action = "RULE16"
endcase

goto current

*****************************************************************
* end of PROC3
*****************************************************************

*****************************************************************
procedure PROC4
parameter start_mark,current,vox,action
```

```
* this procedure determines which rule applies to a sequence
* of 2 quarters; the possible rules are 8 & 9.

select 1
goto start_mark
skip
do while STEP&vox = "    "
   skip
enddo

do case
case abs(val(STEP&vox)) = 0
  action = "RULE17"
case abs(val(STEP&vox)) = 1
  action = "RULE9"
case abs(val(STEP&vox)) >= 2
  action = "RULE8"
endcase

goto current

*****************************************************************
* end of PROC4
*****************************************************************

*****************************************************************
procedure PROC5
parameter start_mark,current,vox,action

* this procedure determines which rule applies to a sequence
* of 8 16ths; the possible rules are 1 & 19. We also check
* for rules 24 and 25 here

select 1
goto start_mark
action = "    "

* first, check if the first note is not on a downbeat.  if it
* is not, we do nothing because we don't want to start
* looking at a series of 16ths starting in the middle of
* something.

if BEAT = "  "
  * need to check previous note for rules 24 and 25
  skip -1
  * rule 24
  if NOTE&vox = "9999"
    action = "RULE24"
    skip
    return
  endif
  * rule 25
  if ltrim(trim(MELODY&vox)) = ;
        "S".or.ltrim(trim(MELODY&vox)) = "A"
```

```
    action = "      "
    current = recno()-1
    no_skip_flag = "F"
    return
  else
    current = recno()+1
    action = "      "
    return
  endif
endif

stepwise = "T"

skip
* check rule 1 first: stepwise 16ths are slurred
cnting = 1
do while cnting <= 7
  if abs(val(STEP&vox)) = 1 .or. abs(val(STEP&vox)) = 2
    stepwise = "T"
  else
    stepwise = "F"
    exit
  endif
  skip
  do while STEP&vox = "    "
    skip
  enddo
  cnting = cnting + 1
enddo

if stepwise = "T"
   action = "RULE1"
   return
else
   * now check rule 19: leap between 1st and 2nd?
   goto start_mark
   skip
   if abs(val(STEP&vox)) = 1 .or. abs(val(STEP&vox)) = 2
     stepwise = "F"
   else
     * there is a leap between 1 & 2
     * but we need to check that the rest are stepwise
     skip
     do while STEP&vox = "    "
        skip
     enddo
     cnting = 2
     do while cnting <= 7
       if abs(val(STEP&vox)) = 1 .or. abs(val(STEP&vox)) = 2
          stepwise = "T"
       else
          stepwise = "F"
          exit
       endif
```

```
        skip
        do while STEP&vox = "    "
          skip
        enddo
        cnting = cnting + 1
      enddo
    endif
endif

if stepwise = "T"
  action = "RULE19"
else
  * since we failed here, we need to try again on the 2 sets
  * of 4 notes
  action = "TRY AGAIN ON 4"
endif

goto current

*****************************************************************
* end of PROC5
*****************************************************************


*****************************************************************
procedure PROC6
parameter start_mark,current,vox,action
* this procedure determines if rule 7 applies to an uneven
* sequence of 16ths

select 1
goto start_mark

skip
do while STEP&vox = "    "
   skip
enddo

* check for stepwise/leaps
cnting = 1
do while cnting <= (current-start_mark-1)
    skip
    do while STEP&vox = "    "
      skip
    enddo
    if abs(val(STEP&vox)) = 1 .or. abs(val(STEP&vox)) = 2
      stepwise = "T"
    else
      stepwise = "F"
      exit
    endif
    cnting = cnting + 1
enddo
```

```
if stepwise = "T"
  action = "RULE21"
else
  action = "    "
endif

goto current

*************************************************************
* end of PROC6
*************************************************************

*************************************************************
procedure PROC7
parameter start_mark,current,vox,action

* this procedure checks if rule 7 applies to an eighth note
* all by itself

select 1
goto start_mark
action = "    "

if BEAT = "  "
  * find out if next beat is 1 or 3 since the accents are on
       * 1 or 3
  do while BEAT = "  "
    skip
  enddo
  if (BEAT = "1" .or. BEAT = "3")
    if RNOTE&vox != "    "
      action = "RULE7"
    else
      action = "    "
    endif
  else
    action = "    "
  endif
endif

goto current

*************************************************************
* end of process.prg
*************************************************************
```

References

Alphonce, B. (1980) "Music Analysis by Computer", In Computer Music Journal, 4(2), p. 26.

Anderton, C. (1986). MIDI For Musicians, Amsco Publications, New York, NY.

Apel. W. (1972). The Harvard Dictionary of Music, The Belknap Press of Harvard University Press, Cambridge, MA.

Arenson, M. (1984). "Computer-Based Instruction in Musicianship Training", In Computers and the Humanities 18(3/4), p. 157.

Baroni, M. and Jacoboni, C. (1975). "Analysis and Generation of Bach's Chorale Melodies", In G. Stefani, ed. Proceedings of the First International Congress on the Semiotics of Music. Centro di Iniziativa Culturale, Pesaro, Italy.

Baroni, M. and Jacoboni, C. (1978). Proposal for a Grammar of Melody, Presses de l'Universite de Montreal, Montreal, Quebec, Canada.

Baroni, M., and Jacoboni, C. (1983). "Computer Generation of Melodies: Further Proposals", In Computers and the Humanities, 17(1), p. 1.

Barr, A. and Feigenbaum, E. (1981). The Handbook of Artificial Intelligence, Volume I, Addison-Wesley, Reading, PA.

Barra, D. (1983). The Dynamic Performance: A Performer's Guide to Musical Expression and Interpretation, Prentice-Hall, Englewood Cliffs, NJ.

Bennett, J.S., Creary, L.A., Engelmore, R.M., and Melosh, R.E. (1978). "SACON: A Knowledge-Based Consultant in Structural Analysis", in Heuristic Programming Report No. HPP-78-23, Computer Science Dept., Stanford University, Stanford, CA.

Bernstein, L. (1976). The Unanswered Question, Harvard University Press, Cambridge, MA.

Birnes, W. (1989). McGraw-Hill Personal Computer Programming Encyclopedia Second Edition, McGraw-Hill, New York, NY.

Bodky, E. (1960). The Interpretation of Bach's Keyboard Works, Harvard University Press, Cambridge, MA.

Bolognesi, T. (1983). "Automatic Composition: Experiments with Self-Similar Music", In Computer Music Journal 7(1), p. 25.

Braider, J. (1980). "Anthony Newman - Renaissance Man of the Classical Keyboard", In Contemporary Keyboard, December.

Brooks, F. (1957). "An Experiment in Musical Composition", In IRE Transactions on Electronic Computers, Vol. EC-6, p. 175.

Brown, J.S., Burton, R.R., and Bell, A.G. (1974). "SOPHIE: A Sophisticated Instructional Environment for Teaching Electronic Troubleshooting", in BBN Rep. 2790, Bolt Beranek and Newman, Inc., Cambridge, MA.

Carbonell, J.R. (1970). "AI in CAI: An Artificial Intelligence Approach to Computer-Aided Instruction", in IEEE Transactions on Man-Machine Systems 11(4), p. 140.

Charnassé, H., and Stépien, B. (1986). "Automatic Transcription of 16th-Century Musical Notations", In Computers and the Humanities, 20(3), p. 179.

Chomsky, N. (1957). Syntactic Structures. Mouton, The Hague, The Netherlands.

Cleaveland, J. and Uzgalis, R. (1977). Grammars for Programming Languages. American Elsevier, New York, NY.

Conger, J.(1988). C Programming For MIDI. M & T Publishing, Redwood City, CA.

Conger, J.(1989). MIDI Sequencing in C. M & T Puublishing, Redwood City, CA.

Cope, D. (1987). "An Expert System for Computer-Aided Composition", In Computer Music Journal 11(4), p. 30.

Crawford, D. and Zeeff, J. (1983). "Gregory's Scribe: Inexpensive Graphics for Pre-1600 Music Notation", In Computer Music Journal, 7(1), p. 21.

Crerar, M. (1985). "Elements of a Statistical Approach to the Question of Authorship in Music", In Computers and the Humanities 19(3), p. 175.

Donington, R. (1982). Baroque Music: Style and Performance, W.W. Norton & Company, New York, NY.

Donington, R. (1973). A Performer's Guide to Baroque Music, Faber and Faber, London, England.

Duda, R.O., Gaschnig, J., Hart, P.E., Konolige, K., Reboh, R., Barrett, P., and Slocum, J. (1978). "Development of the PROSPECTOR consultation system for mineral exploration", in SRI Projects 5821 and 6415, SRI International, Menlo Park, CA.

Ebcioglu, K. (1988). "An Expert System for Harmonizing 4-Part Chorales", In Computer Music Journal 12(3), p. 43.

Green, D.M. (1965). Form in Tonal Music, Holt, Rinehart & Winston, New York, NY.

Gross, D. (1981). "A Computer Project in Music Analysis", In Computing in the Humanities, Lexington MA, p. 299.

Gross, D., and Griffin, W. (1982). "Implementation and Evaluation of a Computer-Assisted Course in Musical Aural Skills", In AEDS Journal, 15(3), p. 143.

Grout, D.J. (1973). A History of Western Music, W.W. Norton & Company, New York, NY.

Haus, G. (1983). "EMPS: A System for Graphic Transcription of Electronic Music Scores", In Computer Music Journal, 7(3), p. 31.

Hill, J., and Ward, T. (1989). "Two Relational Databases for Finding Text Paraphrases in Musicological Research", In Computers and the Humanities 23(2), p. 105.

Hiller, L., and Isaacson, L. (1959). Experimental Music, McGraw-Hill, New York, NY.

Hiller, L., (1970). "Music Composed with Computers", In Lincoln, H. (ed.), The Computer and Music, Cornell University Press, Ithaca, NY.

Holtzman, S. (1981). "Using Generative Grammars for Music Composition", In Computer Music Journal 5(1), p. 51.

Jones, K. (1981). "Compositional Applications of Stochastic Processes", In Computer Music Journal 5(2), p. 45.

Kendall, G. (1981). "Composing from a Geometric Model: Five-Leaf Rose", In Computer Music Journal 5(4), p. 66.

Klir, G.J. (1985). Architecture of Systems Problem Solving, Plenum Press, New York, NY.

Knopoff, L., and Hutchinson, L. (1981). "Information Theory for Musical Continua", In Journal of Music Theory Vol. 25, No. 1.

Kreitler, H., and Kreitler, S. (1972). Psychology of the Arts, Duke University Press, Durham, NC.

LaRue, J. (1970). Guidelines for Style Analysis, W.W. Norton, New York, NY.

LaRue, J. (1988). A Thematic Catalog of 18th-Century Symphonies, Indiana University Press, Bloomington, IN.

Lerdahl, F. and Jackendoff, R. (1977). "Toward a Formal Theory of Tonal Music", In Journal of Music Theory, Vol. 21, Number 1, p. 111.

Lidov, D., and Gabura, J. (1973). "A Melody Writing Algorithm Using a Formal Language Model", In Computers and the Humanities, 4(3/4), p. 138.

Lorrain, D. (1980). "A Panoply of Stochastic 'Cannons'", In Computer Music Journal 4(1), p. 53.

Loy, G. and Abbott, C. (1985). "Programming Languages for Computer Music Synthesis, Performance and Composition", In ACM Computing Surveys, Vol. 17, Number 2, p. 235.

McAdams, S, and Bregman, A. (1979). "Hearing Musical Streams", In Computer Music Journal 3(4), p. 26.

McDermott, D.V. (1981). "R1: The Formative Years", in AI Magazine, Vol. 2, p. 21.

McKean, R. (1983). "Anthony Newmann: Performing 18th Century Keyboard Music", In Contemporary Keyboard, August.

Meyer, L.B. (1956). Emotion and Meaning in Music, University of Chicago Press, Chicago, IL.

MIDI Manufacturers Association. (1988). MIDI 1.0 Detailed Specification, Document Version 4.0, June, 1988.

Myhill, J. (1979). "Controlled Indeterminacy: A First Step Towards a Semi-Stochastic Music Language", In Computer Music Journal 3(3), p. 12.

Newcomb, S. (1985). "LASSO: An Intelligent Computer-Based Tutor in 16th-Century Counterpoint", In Computer Music Journal 9(4), p. 49.

Newman, A. (1985). Bach and the Baroque: A Performing Guide to Baroque Music with Special Emphasis on the Music of J.S. Bach, Pendragon Press, New York, NY.

Newman, A., ed. (1983). "J.S. Bach: Well-Tempered Clavier Book One", In Great Performer's Edition Series, G. Schirmer, New York, NY.

Orchard, R.A., and Tausner, M.R. (1988). "General Systems: A Basis for Knowledge Engineering", In Systems Practice, Volume 1, Number 2, 165-179.

Orchard, R.A., Reese, E.J., and Tausner, M.R. (1988). "On the Foundations of Knowledge Engineering", In Elzas, M., Oren, T., and Zeigler, B. (eds.), Modelling and Simulation Methodology: Knowledge Systems Paradigms, North-Holland, Amsterdam (in press).

Pearce, A. (1982). "Troubadours and Transposition: A Computer-Aided Study", In Computers and the Humanities 16(1), p. 11.

Pople, H. (1977). "The Formation of Composite Hypotheses in Diagnostic Problem-Solving", in IJCAI, Vol. 5, p. 1030.

Quantz, J.J. (1752). Versuch einer Anweisung die Flöte traversiere zu spielen, Berlin. Trans. and ed. Reilly, E.R., as On Playing the Flute, London, 1966.

Rader, G. (1974). "A Method for Composing Simple Traditional Music by Computer", In Communications of the ACM, Vol. 17, No. 11, p. 631.

Roads, C. (1979). "Grammars as Representations for Music", In Computer Music Journal 3(1), p. 48.

Roads, C. (1985). "Research in Music and Artificial Intelligence", In ACM Computing Surveys, Vol. 17, Number 2, p. 163.

Roads, C. (1986). "The Tsukuba Musical Robot", In Computer Music Journal 10(2), p. 39.

Roads, C. and Strawn, J. (1988). Foundations of Computer Music, MIT Press, Cambridge, MA.

Rosenbloom, D. (1990). "The Performing Brain", In Computer Music Journal 14(1), p. 48.

Rosenfeld, A. (1973). "Array grammar normal forms", In Information and Control, Vol. 23, Number 2, p. 173.

Sacerdoti, E.D. (1977). A Structure for Plans and Behavior, American Elsevier, New York, NY.

Shortliffe, E. (1976). Computer-Based Medical Consultations: MYCIN, American Elsevier, New York, NY.

Sundberg, J., Askenfelt, A., and Frydén, L. (1983). "Musical Performance: A Synthesis-by-Rule Approach", In Computer Music Journal 7(1), p. 377.

Suppes, P., Editor (1981). University-Level Computer Assisted Instruction at Stanford: 1968-1980, Institute for Mathematical Studies in Social Sciences, Stanford, CA.

Sussman, G. and Steele, G. (1981). "Constraints: A Language for Expressing Almost-Hierarchical Descriptions", In Artificial Intelligence, 14, p. 1.

Temporal Acuity Products (1989). Music Printer Plus, Version 3.01, Bellevue WA.

Weiss, S., Kulikowski, C., and Safir, A. (1977). "A Model-Based Consultation System for the Long-Term Management of Glaucoma", in IJCAI, Vol. 5, p. 826.

Winograd, T. (1968). "Linguistics and the Computer Analysis of Tonal Harmony", In Journal of Music Theory, Vol. 12, No. 1, p. 2.

Winston, P.H. (1984). Artificial Intelligence (2nd ed.), Addison-Wesley, Reading, PA.

Xenakis, I. (1974). Formalized Music, Indiana University Press, Bloomington, IN.

Yakoo, Y., and Nagaoka, K. (1985). "Computerized Methods for Evaluating Musical Performances and for Providing Instruction Techniques for Keyboard Instruments", In Computer Education, Vol. 9, No. 2, p. 111.

Zapirov, R. (1960). "On Algorithmic Expression of Musical Compositions", In Doklady, Akademii Nauk, 132(6).

Zapirov, R. (1969). "Cybernetics and Music", In Perspectives of New Music, Vol. 7, Number 2, 115-154.