Representing Musical Knowledge

by

Damon Matthew Horowitz

B.A.
Columbia University
1993

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
in partial fulfillment of the requirements for the degree of

Master of Science

at the Massachusetts Institute of Technology
September 1995

Signature of Author [illegible]
Program in Media Arts and Sciences
August 11, 1995

Certified by [illegible]
Tod Machover
Associate Professor of Music and Media
Program in Media Arts and Sciences
Thesis Supervisor

Accepted by [illegible]
Stephen A. Benton, Ph.D.
Chairperson
Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Representing Musical Knowledge

by

Damon Matthew Horowitz

B.A.
Columbia University
1993

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
on August 11, 1995
in partial fulfillment of the requirements for the degree of

Master of Science

Abstract

This thesis presents an approach to building more intelligent computer music systems. Motivated by the shortcomings of previous systems, I maintain that an enumeration and quantification of musical common sense concepts is necessary for the construction of musically intelligent systems. Demonstrating this approach, a computer system for analyzing and generating improvisational jazz solos has been designed as an initial exploration into the quantification of musical intelligence. The system development has concentrated upon the creation of a knowledge representation and control structure as a model of improvisation in this domain. The design of an architecture to organize and manipulate musical knowledge is influenced by the *Society of Mind* approach of [Minsky 86], and the spreading activation networks of [Maes 89]. This document provides a description of the approach, its motivation, its realization in the system design, and an evaluation of this ongoing project.

Thesis Supervisor: Tod Machover, M.M.
Associate Professor of Music and Media

This work was supported in part by the Yamaha Corporation and Sega of America.

Representing Musical Knowledge

by

Damon Matthew Horowitz

B.A.
Columbia University
1993

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
on August 11, 1995
in partial fulfillment of the requirements for the degree of

Master of Science

Thesis Readers:

[illegible]

Certified by ______________________________
Pattie Maes
Associate Professor of Media Technology
Program in Media Arts and Sciences

[illegible]

Certified by ______________________________
Marvin Minsky
Toshiba Professor of Media Arts and Sciences
Program in Media Arts and Sciences

Acknowledgments

I would like to thank my readers, Marvin Minsky and Pattie Maes, and my advisor, Tod Machover, for their support and advice. This thesis had very different goals than the standard technical report or system description issuing from the lab; I greatly appreciate the sponsorship of these professors for my endeavor to pursue the important larger questions in this field. Their individual contributions to this thesis are evident throughout the body of the document.

I would like to thank Eran Egozy and Alex Rigopulos for working with me to produce the inimitable DBX, for patiently enduring my extended complaints about my work, and for offering sane advice throughout.

I also want to acknowledge Brad Garton for his enthusiasm and openness in providing my initial opportunities to do computer music. His encouragement started the long term project which has led to this document.

There are many others whom I would like to thank for more personal support during this period; they will be further acknowledged personally.

Finally, I would like to thank the Media Lab for sponsoring this work.

Contents

1. Introduction

Music is a bug.
- Marvin Minsky

Music is an artifact of cognitive processes. Minsky suggests that music works by exploiting mechanisms in the human brain designed to perform other tasks[1]. The effect of sound patterns on these mechanisms produces an impression of understanding or experiencing which constitutes our appreciation of music. We would like to understand better the mechanisms of music, the ways in which it affects us. In addition, we need to make better tools to help us create and manipulate music. This thesis addresses the question of how to improve our musical tools by exploring the mechanisms of music. The nature of music is discussed in the context of building computer systems with musical abilities.

1.1. Computer Music Systems

This thesis addresses general musical issues applicable to many types of computer music systems, but focuses upon analysis and generation systems[2]. [Rowe 93] presents a detailed listing of computer music systems, categorized according to their paradigm of interaction. Computer music systems are being developed as interactive performing improvisers [Rowe 91; Horowitz 92], traditional accompanists, compositional aids [Rigopulos 94], teaching tools, and new types of instruments [Machover 92], among other things. (A detailed discussion of systems is found in the *Computer Programs* section.)

The work in our Media Lab research group (Music and Cognition) is representative of the state of the art of generative computer music systems. Since 1985, one of the goals of the Hyperinstruments project [Machover 92] has been to use technology to expand the expressive power of musical instruments. Initially, determinations of the system's behavior in response to the user's gestures was made on an instance by instance basis by the system's designers in the context of a composed piece, in order to ensure the musicality of the output. As our group has begun exploring systems for musical contexts that are less predetermined, further understanding of music is required in order to have the system produce musical output. For example, in the DBX system [Rigopulos 94] a user manipulates the course of a continuously generated musical stream within a constrained genre. Currently, much simple musical intelligence of the user is taken advantage of, for he is responsible for shaping phrases and imbuing the melody with a sense of development, etc.; we would like for future systems to take more responsibility for ensuring the musicality of the output, and to offer a wider range of musical possibilities.

1.1.1. Problems: Unachieved Capabilities

The performance of many computer music systems can be impressive, occasionally matching the output expected from human musicians. However, generation and analysis systems are usually noted for several categories of shortcomings in their behavior: a generative system may produce music that is dull or repetitive, lacking in a feeling of direction, oblivious to issues of context, or otherwise "unmusical"; an analysis system might not seem to be listening to subtle (or obvious) features in the input, or it may assign labels to its input that are not musically salient, that is, that do not correspond to the intuitive concepts with which performers and listeners grasp the music; finally, most systems are incapable of dealing with novel input, and lack sophistication in producing increasingly complex output. Current systems are unable to summarize or abstract or develop musical ideas, they can only imitate or randomly permute or concatenate material without understanding the musical effects of these operations. Addressing specific problems, new systems continue to improve and sound incrementally better through the addition of new features and better fragments of sequenced music. However, systems will continue to suffer major limitations as long as they are not based on a solid working theory of music. Fundamentally, current systems do not understand the music they are generating or analyzing; they lack musical knowledge and musical abilities which are common even to inexperienced audience members. This set of knowledge and abilities constitutes an intuitive definition of *musical common sense.*

A brief discussion of our DBX system [Rigopulos 94] should clarify the nature of the problem. The DBX system consists of a set of simple parameters (which describe the syntax of repeating bass and chord patterns found in a specific sub-genre of popular music), an analysis module (which converts musical lines into this parametric representation), a generation module (which produces musical lines from this parametric representation), and a variety of input control modules (through which users change the parameters in order to influence the output music). One mode of operation of the system is as an automated accompanist to a keyboard soloist; the system analyzes an input solo, and uses this analysis to transform the parameters guiding the output accompaniment.

While the DBX system succeeds in demonstrating the potential of such interactive parametric systems to entertain and educate users within a popular genre, it has several major limitations. The syntactic emphasis of its limited representation (which excludes acknowledgment of concepts concerning the effects or roles of the notes it is generating) results in arbitrary and frequently unmusical transformations of parameter sets. While we would like to be able to command the system to transform music according to an adjectival description (e.g., make it weird or funky, more active, longer, expansive, sad, smooth, etc.), the system lacks the ability to judge its own output to determine if the desired effect has been achieved. Similarly, users of the system easily notice when it has played something unmusical, either of its own accord or through their direction, while the system itself does not even contain a representation in which such evaluation could be expressed. In addition, the system cannot "play itself": it requires a human to steer its output over time, to give the music a sense of direction or evolution, to avoid having excessive repetition, to make its output comprehensible. Nor can it produce common types of variation or ways of responding to input material, such as playing related rhythms or developing melodic ideas, for it has no notion of what a phrase is and how it is structured, the roles of parts, their relative importance and types of stress. In sum, while the system is able to produce output that superficially fits within a genre, it is missing many of the common sense concepts about music that are necessary in order to empower more acceptable and sophisticated behavior.

Creating a piece of music that is similar to another, or recognizing this similarity, is a complicated task that is difficult for current systems, as the following example indicates. Generation of similar material can be roughly approximated by simply maintaining some surface features of the original piece of music while changing or adding others. In the DrumBoy [Matsumoto 93] system, a user enters a drum pattern into the system and then commands the system to transform the pattern according to an adjective label description (for instance, to make the pattern more "active"). To accomplish this, the system removes some fragments of the original pattern which are labeled as particularly inactive (perhaps because of the number of notes used, or type of drum sound they are played on), and adds new fragments which are recalled from a database of patterns hand-labeled as "active". The result usually accomplishes its goal: the user receives a new pattern which is similar to the old one except for being more active. However, the system does not always work: frequently the system will produce something that is musically illogical, such as a conflicting polyrhythm; the system's transformations end up sounding very similar because they are all based upon the same database; and the salient aspects of the input pattern are often removed by the system in the course of the transformation. These problems arise because the system lacks a rich representation of either the patterns themselves or of the very concept of activity. There is an important difference between knowing what activity is, and simply knowing examples of active patterns. The former type of understanding is able to generalize, relate to other features, and be applied flexibly, while the latter can only demonstrate occasional accidental success.

1.1.2. Explanation: Missing Knowledge

One cause for the problems plaguing current computer music systems is that musical material is often treated as though it were sheerly numerical data; that is, it is removed from the context of the musical genre within which it is meaningful, separated from the processes of music cognition that humans use for playing and listening. The numerous attempts to apply numerical learning algorithms (such as neural networks [Todd 91; Feulner 93], genetic algorithms [Biles 94; Horowitz 94a; Horner 91], statistical pattern matching [Cope 91]; Markov models [Zicarelli 87]) to isolated features of music in order to analyze or generate high-level concepts about the music demonstrate this problematic approach. These programs ignore a vast amount of knowledge that we already have about music from both common sense and music theory; their reduction of music to a simple signal, by focusing on a single numerical representation, loses important information. The results of these attempts reveal that these algorithms are better suited to solving specific sub-problems within the context of a larger system. While it is fascinating to observe what concepts can in fact be learned from this minimal amount of information, further progress in musical intelligence requires that we capitalize on the musical knowledge we have by designing symbolic systems which attempt to quantify common sense and music theory concepts. Once this has been accomplished, we may benefit from the experience afforded by the above works and apply numerical learning algorithms to our enriched data, allowing exploration of concepts beyond our current grasp[3].

Several systems have begun to deal with the "knowledge problem" by encoding fragments of knowledge about a musical genre, in an attempt to reduce inappropriate behavior. For example, it is common for generative systems (such as the DBX system) to have a variable representing the currently active "key", such that wrong pitches (ones inexplicably and unjustifiably outside the set of conventionally acceptable pitches) can be avoided. However, these fragments of knowledge are usually isolated, lacking a network of relations to other knowledge. Thus, there is no notion of the effects of or the reasons for rules guiding pitch choices; as a result, the fragments of knowledge are unable to contribute to a deeper model of music. Most of the encoded conventions tend to address questions of surface elements, ignoring both the important structures in music, and the nature of the processing mechanisms which analyze and generate it. The many uses of expert systems to perform a specialized function of analysis or generation fall into this category [Ebcioglu 92; Schottstaedt 89]. These systems are extremely useful for testing the sufficiency of the rules they encode, but cannot develop into more flexible and intelligent systems because they are restricted to a few simple features in the music, and one method of processing information (the production rule). Even systems which claim to have a representation of the "structure" of a musical piece frequently only refer to a grammar rule which states the archetypal form of the music; usually these are unexplained rules, failing to represent the functional purpose of their constituent structures. However, it is exactly this type of functional information which is essential in understanding the music's submission to (or transcendence of) a given form.

I propose that the limitations of contemporary computer music systems result not simply from an absence of more features (which could "patch" their shortcomings), but from a fundamental misdirection in their design. A general evaluation of current systems reveals that most are designed to perform a specialized function through the encoding of a few rules about the relevant human knowledge of music. Current systems address superficial features in music, enumerating rules to handle various surface situations, while it is the structures and mechanisms of music that constitute the understanding which is required for intelligent behavior. Our complaints and frustrations with current systems result from dissatisfactions in our listening processes that learned conventions are not being respected; thus, in order to better satisfy these processes, the processes themselves and the structures they work with should be modeled directly[4].

1.2. My Approach

1.2.1. Musical Intelligence

In order for music programs which emulate human musical abilities to behave in an increasingly intelligent fashion, they must share humans' common sense understanding of music. The long term goal is for a system's knowledge representation to correlate as closely as possible with human ideas about music (insofar as we can articulate descriptions of our understanding of music)[5]. Musical concepts such as "phrase" and "meter" must be represented in terms of their inherent structure, their interconnected relationship to other concepts, and the mechanisms that give rise to their production and recognition. The structures and relationships must reflect our theoretical knowledge of music, while the mechanisms must demonstrate the properties that music perception and cognition research have revealed. I consider a representation of musical common sense concepts combined with the mechanisms that manipulate them to be the basis of *musical intelligence.*

Musical intelligence is that functional understanding which permits a behavioral facility to produce or consume artifacts of a musical culture. A musical culture can be viewed as a large set of concepts about the rules, conventions, practices, and history of a genre. Several concepts are common to many cultures, such as a regular beat, a melody defined by motives, development of material over time, etc.; therefore, some aspects of musical intelligence are applicable to music in general [Jackendoff 87]. Musicians and listeners comprehend music in relation to a genre. An obvious example is a major scale, which is numerically represented as simply a sequence of pitches, but within a traditional western tonal musical culture is understood to be rich in implications, references, rules, tendencies, and dependencies. We would like for our music systems to share this understanding.

Common primary abilities demonstrating musical intelligence are recognition and reproduction of music[6], tasks which require comprehension of a genre's materials and procedures. We consider a system to be demonstrating musical intelligence when its analytical or generative decisions are explicable according to our own common conception of the music. Human level musical intelligence is extremely complex, involving a myriad of processes manipulating multiple representations [Minsky 92; West 91]. The endeavor of building musically intelligent systems consists of suggesting and testing hypotheses about theoretical processes sufficient for explaining the information that is commonly identified in music.

1.2.2. Representations of Common Sense

Until our machines have access to commonsense knowledge, they'll never be able to understand stories -- or story-like music.
- Minsky: [Minsky and Laske 92]

I share Rowe's frustration with previous systems' "algorithmic inability to locate salient structural chunks or describe their function" [Rowe 93]. Reasonable manipulation, abstraction, and variation of music is possible only if the salient features of the music are known. By listing features as separately as possible, and then constructing a network which represents their relations, it is possible to understand musical structures: reductions, interpretations, groups, gestures, and similarities all arise from the relative emphases of the features on the musical surface interacting with a conceptual network of musical knowledge and processes.

A musically intelligent system is able to notice and produce instances of those concepts which constitute humans' understanding of music. I believe that this requires the enumeration of these concepts for explicit representation in the system; knowing the properties of each musical concept when transformed is possible only through listing their connections and constraints. I adopt Minsky's suggestion that the long term goal is a musical common sense database [Minsky and Laske 92], just as many problems in other domains of artificial intelligence require a real-world common sense database [Lenat 90]. Such a database formalizes the properties of musical elements such as "melody" and "variation". The low level terms of this database constitute a framework on top of which high level ideas about emotional qualities, narrative or other referential structures, categorizations of styles, etc., can be explored, and eventually added to the database.

The construction of such a database consists of collecting information about the domain of music, and designing a representational framework that can hold this information in such a way as to make it amenable to musical processing. This thesis advocates the maintenance of multiple perspectives and representations of musical material, with explicit interconnected relations of each instance of music to the network which constitutes the system's musical knowledge. This approach allows for the manipulation and comparison of musical instances to be performed in consideration of the appropriate features for each desired functionality. This general theoretic approach will become clearer as the processes which work with these representations are explained in the *Processing Music* section.

1.3. An Intelligent System

The present goal, then, is a computer system that knows how to calculate those obvious things about music which constitute our common sense understanding. Towards this end, I have created a model of jazz improvisation which reflects my approach towards musical intelligence. The model is a design for a computer program that contains explicit knowledge about melodic line and conventions in a specific genre. Listening tasks have been emphasized in the current work because they directly reflect our musical sensibilities, namely our ability to recognize and understand music[7].

The difficulty of this work is the translation from intuitive ideas about how music works to quantified mechanisms which reflect these ideas. This is not merely a difference in level of detail; it is the difference between a speculative and a substantiated theory. There are an infinite number of intuitively appealing or plausible propositions which can be made about the nature of music, and we are in fact constantly barraged with suggestions that "music is about similarity," or "music is about surprise," etc. But the question of what constitutes musical similarity or surprise is left undefined, and is in fact the heart of the problem. The quantification of a number of such intuitions into a consistent theoretical framework is the design work of this thesis.

The set of specific capabilities which this system was designed to express were decided upon as representative of human common sense processing. Primarily, the system is designed to know about features in the musical surface and the ensuing structures which they create, in terms of being able to recognize, produce, and compare, and transform these structures. This knowledge consists of an embedded representation of harmony and meter, of song form, and of abstracted representations of a melodic line (summarized in the*Relation to DBX System* section). This understanding defines a similarity metric for instances of music, which is vital towards knowing how to create variations, or how to notice the references and parallelism of a line (the semantics of musical structures). As a result, the system can find similar patterns in lines, or compare solos and understand the types of differences between them. In addition, the system knows what constitutes conventional behavior in the genre; thus, it can understand things that are novel and within this category, and also notice things that are strange or problematic. Related to this capability is the ability to recognize the qualitative effects of surface musical elements in terms of their relation to the set of known musical constructs. The system is able to evaluate the effect of combining or sequencing phrases or fragments of lines to determine the compound musical quality, which is not always simply the sum of the parts.

1.3.1. Products of Musical Intelligence

This basic set of capabilities, elaborated upon and explained in detail in the body of this document, constitutes the initial model of musical intelligence which the system expresses. While this is only a beginning, it is already a large step towards several impressive qualitative abilities or applications which reflect musical intelligence. With this framework, we have the terms in place to experiment with models of solo construction, to reflect a narrative structure or a sense of direction and return. We can also explore different parameters of comprehension of music, such as the factors which contribute to expectations and predictions, and the factors which result in confusion. On a more basic level, the system is enabled to understand what is musically acceptable within the studied genre, to know the constraints, freedoms, relations, and roles of parts of a melodic line. This understanding is necessary for a system to behave in a reasonable fashion; it is this type of understanding which the DBX system needs in order to offer intuitive forms of control to users while avoiding mistakes.

In order for a computer system to respond to a human, to interact, to do what it is intended to do, and to be able to maintain this ability in a flexible and extendible manner, it must share the human's knowledge of the domain. The only way to explore issues of musical affect, to understand the exciting or jarring effects of music, is to know the conventions which these are operating against. For instance, this system is able to understand the emphasis of a syncopated peak in a phrase, it knows how this can be manipulated and transformed to relate to similar items in different contexts. The applications of systems with increased musical intelligence are wide and varied; these are discussed in *Appendix 1.*

1.3.2. System Functionality

The utility of the new system can be understood as extensions to the functionality of the DBX system. The new system provides a representation of and processes for finding the following features: similarities, and the ability to exhaustively compare and account differences (e.g., of rhythms or contours); parallelism and reference; metric forms and the relation of the surface line to the meter; structures larger than local note decisions, including reductions and groupings, and the relations of these to surface features; contour features and similar regions of contours; conventions for solo structure, such as where repetitions usually occur, and possible successions of phrases related by symbolic labels (such as affect); conventions for relating to harmony and meter; deviations from conventions, such as apoggiaturas and syncopations; relative emphases of successive musical elements; cross rhythms, as a form of alternate metrical grouping as well as a rhythmic pattern of emphasis; and color characteristics of regions, harmonically and in terms of rhythmic categorizations.

1.4. Contributions of this Thesis

This project was conceived of as a doctoral project on musical intelligence, entailing the building of a musical common sense database, and the demonstration of its use in an intelligent jazz improvisation program. The current thesis has begun this project by creating the representational framework for such a database. The work on the project to date can be categorized into theoretical, system design, and implementation components. The point of this thesis is theoretical, proposing an approach that can be taken towards building better music systems, and a perspective on thinking about music and mental processing. This is explored through a tangible domain and task, wherein a detailed model has been designed and implemented. The main work of the thesis has been the translation of theories and intuitions into developed, refined, quantified, and consistent elements of a computer system. These contributions are summarized below:

• Theoretical: the analysis of the limitations of current systems, and the determination of the underlying problem; the formulation of a new approach towards their general improvement.

This approach is summarized in the above introduction, and described in more detail in the *Understanding Music* section.

• Design: the application of my approach to the design of a system for jazz improvisation.

The steps of this process were: the determination of information/knowledge present in the domain (the structure of the domain "semantics') necessary for intelligent behavior; the quantification of this knowledge into a set of explicit representations, indicating how the information is organized, accessible, manipulatable; the construction of processes using these representations for listening and generation. This is described in the *Building a More Intelligent Music System* section.

• Implementation: the partial implementation of the current design; the implementation of related projects towards this goal.

The implementation of the design completed to date includes all the main modules described in the *Representing Music* section of this document, most of the listening functionality, and the architecture/utilities that support these; the *Implementation Status* section describes the current status of the program.

The related projects completed during this period use simplified representations in several contexts: a spreading activation network for generation, using an earlier version of the design; analysis and generation modules for melody and rhythm, and a *Response State* framework for representing the relationship between a soloist and accompaniment, in the DBX system; and an improved rhythmic representation for the GRGA system, on top of which a genetic algorithm learns user's preferences for percussion textures [Horowitz 94a].

1.5. Document Outline

The *Understanding Music* section is intended to present a collection of intuitive and theoretical ideas about how musical understanding works. In this section, I step back and discuss the general nature and definition of musical understanding from the point of view of music theory, from artificial intelligence, and from my own experience as a musician. I discuss the influence of music theory on computer music systems, the implicit representations of musical knowledge in these theories and systems, and how these have impacted my formulation of a new approach.

The *Building a More Intelligent Music System* section then discusses the specific model and system that I developed based upon my approach. This system is both a reflection of the ideas presented in the *Understanding Music* section, in that it refines, quantifies, and implements some of these ideas, and a framework on top of which the more difficult ideas can be explored. I define the restricted problem domain and task that I am working towards, the representations of music that I have determined are necessary, the design of the processes that produce behavior using these representations, and then the implementation as it currently stands. Note that the *Representing Music* section is the heart of the thesis, where the actual representations are described. The rest of the document explains the motivations for and utility of these simple structures.

2. Understanding Music

The interest of this thesis is to investigate musical intelligence through the construction of a system which demonstrates intelligent abilities. This section outlines the theoretical framework which guides this endeavor; this discussion is alternately formal, in describing a previously defined specific mechanism, and intuitive, in describing the common sense ideas which it is our goal to quantify. First, my use of the term "understanding" is briefly clarified, emphasizing a summary of the theory of mind presented in *Society of Mind* [henceforth SOM: Minsky 86]. Next, musical knowledge is characterized in order to give a picture of the domain to which this understanding should be applied. The next sections present a variety of perspectives on musical understanding in terms of musical knowledge; previous work related to this area is summarized. The final section provides a summary presentation of intuitive theories of musical processing. These sections together constitute an expression of musical understanding, the goal for an intelligent system. The work of the thesis has been split between the formulation of this goal and the actual design of a system (described in the *Building a More Intelligent Music System* section).

2.1. What is Understanding

2.1.1. Relation to Artificial Intelligence

Artificial intelligence is a widely used term, loosely referring to a broad range of theoretical and engineering research. My interest here is in those portions of the discipline which involve the direct study of knowledge representations and control structures necessary to produce general intelligent behavior. This project benefits from the extensive and diverse explorations which this sub-field of artificial intelligence has conducted over the past forty years, and inherits most importantly a general sensibility for approaching these problems. This section briefly mentions general work in symbolic artificial intelligence that has influenced this project.

The development of concepts of scripts, conceptual dependencies, plans, and frames [Schank 77, 75; Minsky 75; Agre 88] all reflect a generalization of theoretical analyses of the information in a domain and the processes which operate on that information. These paradigms are relevant influences here because they suggest models through which to interpret intelligent behavior in a general sense that can be applied to new domains. Similarly, viewing common sense as a crucial knowledge-intensive aspect of intelligence (a perspective literally adopted by the CYC project [Lenat 90]) is a widely applicable idea, one which I believe will be fruitful in the music domain; this motivates my interest in developing appropriate representations and management procedures to enable meaningful access and manipulation of stored musical material.

More recent work in behavior-based, or agent-based, artificial intelligence suggests an alternative perspective from which to view the construction of intelligent behavior [Maes 94]. The use of spreading activation networks for action selection provides a robustness and flexibility unavailable from more traditional rule-based approaches [Maes 89][8]. However, much of the precision and expressive power of traditional methods is lost with these approaches, and there is difficulty in scaling them to larger and more cognitively challenging domains. Therefore, I work with a hybrid of traditional and agent methods [Horowitz 94b]; this is appropriate because music is a domain which presents a dynamic and unpredictable environment combined with the need for rational high-level structural behavior.

2.1.2. Minsky and Understanding

The work in artificial intelligence which has the largest influence over my approach is the theory of mind described in SOM [Minsky 86]. Minsky suggests that intelligent behavior can arise from the interaction of many specialized relatively unintelligent smaller agents. Each of our high-level actions and thoughts is described as requiring the activation of many different kinds of representations and processes with localized information. The impression of the SOM approach is evident throughout the description of my system; for example, my approach literally adopts the SOM use of agents to separate functionality and *K-lines* to organize memory structures.

Minsky describes understanding as a way of referring to the complex relationship between a set of processes and a database of information. Understanding consists of the multifaceted ability to manipulate information correctly: to relate new information to old information, to form associations, to create psychological packaging that reflects the meaning of the information such that we can label it and explain it. Information is only useful knowledge if it is described in terms of the relationships between things and their potential uses. Something is effectively understood when we have sufficient ways to manipulate it that it becomes useful for our purposes. Minsky further suggests that through forming abstract reductions of instances and categorizing them we can increase our understanding; thus a representation of information should reflect reductions in order to take advantage of shared features between different concepts.

SOM presents a description of language (the ReDuplication theory) in terms of understanding. I believe this description is roughly applicable to music as well, so this paragraph briefly introduces some of these concepts of language processing. Minsky suggests that language is intended to communicate meaning by creating a structure in the listener's mind which reflects the meaning of the intended structure from the speaker's mind. The following mechanisms of language are used to direct the listener's attention to the right structures in his own mind, and then to guide the association of these with each other. As speech is heard, *nemes* from different realms compete to assign their concepts and terms as the interpretations of the signal; the shifts in meaning of a phrase are the result of the relative strength of different nemes. A single word in language is a *polyneme* which touches *K-lines* in different agencies to set them to the correct state to reflect their sense of the word's meaning. *Pronomes* are similar to pronouns in language grammars; they represent roles which serve as bins to which the constituents of a phrase can be temporarily bound in creating a string of attachments. These roles are the learned constituents of phrases, which are the basic structural unit which we find in language and music (and even vision Minsky suggests). A comprehensible speaker, therefore, is one that provides a sequence of tokens which fit the expected sequence of roles that our pronomes are looking for. However, individual word-nemes can also skip the grammar frames and directly activate story frames or other structures to communicate meaning.

My interest here is not in a literal mapping of a language theory to a music theory, but rather in using a linguistic theory to inspire and motivate perspectives on a music theory. The equivalent of a word in music can be seen as any number of musical tokens, such as a note, a motive, or a phrase. The important idea is that the word is an instance which is sensed by a structure that is building forms and meanings between words. Thus we apply the spirit of this language description to music by describing music processing as the sensing of units which fill the slots in higher level forms; the message that is transmitted is the combination of the specific tokens' associations and the high level form, while the nature of the transmission is determined by how well the sequence of tokens matches the forms we have available for understanding musical sequences. Thus, musical knowledge needs to reflect the implications of instances of music in terms of the structures which they activate.

2.2. What is Musical Knowledge

Given the above description of understanding, this section describes the type of information and processing which constitute musical understanding in particular. Subjective reports of introspection about musical knowledge vary wildly, ranging from memories of how it feels to be at a concert, to recollections of particular harmonies and melodies, to descriptions of small details of how certain musical problems are realized and solved [Minsky 89]. These reports can be seen as samplings of the information present at different levels of abstraction; part of musical knowledge, then, is knowing how these abstractions are defined.

Intuitively, musical knowledge reflects the salient properties of an individual's experience of music. The categories, generalizations, and relationships of these properties make this information *knowledge* rather than simply unorganized lists of facts. For a particular genre, musical knowledge describes the usual properties of the musical surface and structures, which can also be viewed as the conventions for the musical behaviors which create the music [West 91]. An intelligent listener understands these properties in terms of their relation to other properties, which define the reasons for their existence, their meaning.

The meanings evoked by musical instances are references to other musical instances, both in terms of literal similarity and in terms of similarity of tone or category. Musical knowledge includes representations which define our similarity metrics for music. For example, music cognition experiments [Dowling 94] have shown that melodic contour is a salient feature in the recollection of simple tunes. These types of experiments demonstrate the music is not understood simply as a list of pitches, but as a complex phenomenon which includes reductions of the surface into descriptions of features. The representations of these features, in addition to the surface, can then be used to make similarity judgments. Any single representation will be insufficient to explain widely varying types of similarity which humans can readily perceive; this is demonstrated in detail in the *Example Comparisons* section.

Thus, musical knowledge involves the construction of multiple representations to reflect the different features of music which are relevant for different purposes. The separate levels of these representations are invoked for different processes when listening to music. We commonly listen to music that we have heard before, and still enjoy it. Perhaps this is because representations on a low level are still stimulated in the same way to cause excitement and expectation regardless of the knowledge the high level contains about how the structure will inevitably unfold (see *Processes of Listening* section).

2.3. Previous Representations of Musical Knowledge

A computer system or a theory of music is a proposed representation of musical knowledge, a suggestion of how understanding may be explained in a domain. Music theories[9] usually have the shortcoming of not being quantified, while computer programs usually have the shortcoming of not being motivated by a good theory. In this section I will briefly mention prominent music theories, cognitive theories of improvisation, previous efforts to build jazz systems, and more general music systems whose approach is related to mine in representations of knowledge or use of artificial intelligence techniques. This section simply summarizes the relevant contribution from these areas; the relation to my work is made clear in the following sections.

The collection of disparate ideas presented here forms the set of previously developed knowledge fragments from which I have attempted to assemble a functional theory and model. The limitations of the computer programs motivate my approach to build better systems, while the incompleteness of the music theories inspires my approach to quantify these theories. Inspired by the potential of these ideas, I have quantified and refined them, and constructed a framework on top which they can be explored. I have added to this set my own observations and analyses, which are included as commentary in these sections and as the main subject of the *Intuitive Representations of Musical Knowledge* section.

2.3.1. Music Theory

Minsky criticizes traditional music theory by suggesting that it focuses only on nomenclature, "classification of chords and sequences", thus ignoring the more important "semantics" of music processing [Minsky 89]. We would like to have good theories about how music functions, how it uses reference and artifacts of our processing to create a sense of meaning, to provide us with the experiences that we have in reaction to it. Further, we want these theories to be quantified to a degree that they can be tested with computer programs, rather than remaining speculative.

However, despite these criticisms, music theory as it stands does still offer useful material for this project in two ways. First, the conventions which it enumerates are important to know in addition to the processes that create them. Second, the more speculative sides of these theories can serve as inspiration for the design of more quantified theories, which result from refining past theories and testing them in actual designs for algorithms. The sections below present music theory perspectives that are relevant to the project for these reasons.

Meaning and Affect

In the discussion of meaning in music, [Meyer 56] classifies music theorists as being either absolutists (the meaning is in the musical relationships) or referentialists (meaning in associations to the world), with the further categories of formalists (interested in intellectual appreciation) and expressionists (that music, from either point of view, evokes emotions). In any case, he argues that there is abundant evidence (from generally accepted introspection) that music communicates something emotional[10], and thus has some meaning. Meyer's well known contribution to the study of this meaning is the proposal that affective state arises when an aroused urge, nervous energy, or a tendency to do something is frustrated: it is repressed, not satisfied, or unable to come to fruition. This suggests the implication-realization theory that musical affect arises from disappointed expectations.

Narmour's theory of melody construction [Narmour 90] relates to this general theory, by presenting a detailed description of the possible types of melodies in terms of the implications their constituent notes present, and the degree to which the ensuing predictions are realized. Narmour categorizes melodic motion into eight archetypal forms, based upon the relation of successive step sizes and directions. One simple and important idea stated here is that smaller steps suggest their continuation in similar small steps, while leaps suggest a reversal, wherein the next note is a step in the opposite direction (termed *Gap-Fill*). His calculus of melody allows for the definition of levels closure in a melody ("articulative" of low-level melody, "formational" suggesting higher level, "transformational" showing a new portion of hierarchy), defined by standard conventions (e.g., consonance resolves dissonance, duration moves cumulatively from short to long, metric emphasis is strong, an interruption of patterning, change in direction, decrease in size of interval motion). Summing up these ingredients, he then describes implication as essentially non-closure.

From an entirely different perspective, [Clynes 78] discusses the idea of "essentic forms" as dynamic forms with innate meaning. He claims that these forms "act upon the nervous system like keys in a lock", activating specific brain processes of emotion. Clynes presents a radical approach to the discussion of affect; he performs extensive experiments in which finger pressure over time is measured as an expression of the subject's emotional state, and the resulting dynamics are applied to analysis (and construction) of musical compositions. Clynes' work is mentioned here because it is a remarkable attempt to directly quantify emotional concepts and their relation to music. However, much of the ensuing discussion is unverified and overly simplistic; it remains a future project to attempt to apply his ideas to the enriched representation of music developed here.

Language and Grammars

The study of natural language is of interest here insofar as it provides a much studied perspective on how to quantify aspects of communication, as seen above in the discussion of language in SOM. The parallels between music and language are many, for both concern the general topic of human utterance, though there are also significant differences (in particular, the difference in concepts of semantic reference). Particularly interesting to this project are investigations into prosody and intonational cues used in communication, which may suggest actual features of language and music signals that are processed similarly and thus result in similar meanings. For example, [Pierrehumbert 90] describes the importance intonational contours in discourse analysis. Since music as well as discourse has a phrase structure, in which successive phrases are seen as atomic units in relation to each other, the intonational cues which are used in language may be applicable to musical structures as well. From a different perspective, [Scherer 84; Sundberg 82] present evidence about the utility of the acoustic, or intonational, properties of speech in the communication of affect. Again, to the degree that music is heard as though it were language, these ideas are immediately relevant to musical understanding.

Grammatical parallels have also been explored, and are the basis for one of the most influential contemporary books on music theory, *A Generative Theory of Tonal Music* [henceforth GTTM; Lerdahl and Jackendoff 83]. GTTM attempts to describe the intuitions of an experienced listener in a musical genre. These musical intuitions are described largely by preference rules (based on cognitively motivated and linguistically conventional methods) with which structures are extracted from a musical surface. The resulting structures include: *groups*, which are containers of contiguous notes (possibly overlapping) which are similar in features or proximate; *metric structure*, which refers to the common perception of a regular pulse of alternating strong and weak events on successive levels of a hierarchy; *timespan reductions*, in which notes in a group are shown to be elaborations of a head note; and *prolongation reductions*, in which notes are sorted according to which of their adjacent stronger notes they progress towards or are a prolongation of. One reason why this work is significant is for its attempt to detail musical structures; these structures are important aspects of musical comprehension, recognized as being matters of common sense, but have not been explicitly enumerated previously. However, GTTM has been criticized for ignoring the mechanisms which might relate to its structures, and thus being unmotivated in terms of not having an explanation of what roles these reductions might play in the larger processes of musical understanding [Minsky 89].

As with many music theories, GTTM is also problematic because its rules are too imprecise to be quantified, and thus it does not actually offer a complete generative theory. Further, many of the terms it uses are not sufficiently defined, such as "parallelism" and traditional "harmony". This reflects the general sense in which this theory is intuitively appealing, but requires further refinement. As seen below, the system I have constructed utilizes these intuitions, but quantifies them and focuses upon the processing mechanisms as well as a description of the surface and resulting structure. For example, an explicit definition of parallelism through similarity is defined. In addition, my system does not try to create a single theoretically correct reduction, but rather provides processes with which competing interpretations can be created.

The structures and groupings implied in musical surfaces lead to segmentations of phrases, which are essentially labels for levels within hierarchical reductions such as those of GTTM. [Lidov 75] surveys several theorists' discussions of groupings and rhythmic perception, attempting to find categories of groups (such as beginning/middle/end accented, or listings of poetic feet and their recursion). The utility of such categories is their ability to allow formulation of rules about phrasing, such as the use of end-accented groups for musical endings. (A initial quantified description of more simple phrase-finding ideas is presented in [Fraser 93], using durational contours, metric/harmonic stability, and repetition to determine possible segments.) Labeling himself a "structuralist", Lidov is searching for meaning in these structures, rather than assuming that common forms (such as eight bar structure in a sonata) simply have a statistical explanation. Most importantly, he explicitly states that contemporary theories must acknowledge ambiguity and multiple interpretations, using theories of rhythm perception to emphasize the possible ambiguities.

Cognition and Improvisation

Within cognitive science, the fields of music perception and music cognition seek to identify and experimentally verify properties of human musical processing [Handel 89]. Relevant here are the results found regarding mental concepts of tonality, rhythm, and melody [Bharucha 94; Desain 92; Dowling 94;

Krumhansl 90]. These results provide initial models for representations that reflect musical features with proven psychological salience; for example, memory of music appears to rely upon strong concepts of tonality and periodicity while emphasizing features of contour and harmonic distance. Refining these ideas such that they can be implemented allows for their evaluation computationally, which often reveals problems that are overlooked experimentally[11].

Cognitive theories of jazz improvisation attempt to suggest cognitively plausible processing paradigms that can explain observed musical behavior. [Johnson-Laird 91] presents some fundamental ideas regarding the difficulty in obtaining information about improvisers' processes which are not available to their conscious introspection. His discussion focuses upon the use of grammars, suggesting that "tacit" grammars exist for rhythm fragments, and that finite state automata are likely encodings for pitch sequences. He also hints at a useful distinction between passive and generative knowledge, including conscious and sub-conscious perception of harmony. Although he provides an interesting discussion of general concerns, his complete separation of rhythm and pitch processing, and general reduction of melodic information to solely pitch contour, are too oversimplified to be generally useful.

[Pressing 85] provides a more comprehensive initial approximation of some of the psychological elements that an improvising system should accommodate. He views improvisation as consisting of moment to moment choices to continue or interrupt (or add ornamentation, displacements, or phrase closure templates) a current context, as established by the associations of the previously played material; computations of new material are constrained by the previous referent material, goals of performer, and stylistic norms. He begins to express the possibility of abstracted elements of previous contexts resonating in activation networks on several levels, an idea concretely discussed by Minsky [Minsky 86], enabling a concept of direction to emerge as the change of abstract features over time. The need for multiple and redundant representations is noted by Pressing, who also points out the formulaic aspect of improvisation, and its reliance upon traditional themes and techniques. The interaction between high cognitive and low motor action is elaborated here also (noting the constant shift in control in the active improviser), as is the importance of feedback in this system for making controlled motor processing become automatic when learning to improvise.

This final perspective on improvisation is discussed most thoroughly by [Sudnow 78], who literally places the responsibility for many jazz processes in the kinesthetic or motor-memory of the musicians' hands. This book is an elaborate meditation and introspection on his own process to learn jazz improvisation: his fundamental point is that his hands' learned movements contain much of the knowledge that he has available to him as an improviser. This is useful here both as a literal suggestion of a type of processing that should be encoded, specifically motor pattern sequences accessed by "type of thing" desired to play, and also as a testimony to the difficulty in obtaining information about the processes which an improviser uses.

Jazz

Unfortunately, there is a dearth of solid analytic material about jazz available, relative to the treatment of classical music. The syntactic conventions of jazz have been enumerated in several studies, such as [Amadie 90; Mehegan 59]; traditional music theory concepts are simplified here into listings of modes and the ensuing "tension and release tones". In addition, some of these works express less precise notions about the nature of improvisation [Baker; Reeves 89]; despite their vagueness, these ideas still provide inspiration and orientation in designing high-level constructs for my system. Similarly, historical and biographical studies [Schuller 68; Collier 83; Berliner 94] usually contribute interesting impressionistic commentary, but incomplete analytical theories.

Schuller's descriptions are valuable for their historical thoroughness and their enumeration of conventions, as well as the uninhibited enthusiasm for the music which they reflect (see in particular his ecstatic description of Louis Armstrong's *West End Blues*, or *Big Butter and Egg Man*). He convincingly describes several syntactic features of Armstrong's solos, and notices their progression over time, while observing a corresponding progression in the subjective quality of these solos. He describes "tiny phrase cells" which begin to recur in Armstrong's early work, and later become the pivotal stand-by devices of his solos: a descending ninth arpeggio followed by a leap of a seventh; three repeating quarter notes starting on a downbeat; extended cross rhythms with a heavy swing; a short note followed by a much longer repetition with a pronounced vibrato; double time breaks with slight rushing in the middle, starting with a rip to a high note. Relying on the use of these devices, Armstrong frees himself from the New Orleans tradition of melodic elaboration and begins to create original inventive tunes in his improvisations. Schuller frequently uses descriptors such as "daring syncopation", "tonal ambivalence", "fertile tension", "haunting simplicity", or "stammering repetition" in his colorful commentaries of Armstrong's playing. While these clearly evoke a meaning in human readers, their quantified definition requires work such as the present thesis in order to explain exactly what is meant.

Individual research on particular artists [Owens 71; Kernfeld 81; Smith 83] provides a greater depth than other analytic discussions about jazz, but is frequently limited in analytic perspective. Owens transcribed hundreds of Charlie Parker solos, and came up with one hundred "primary motives" from this set, defined as anything from a few notes to a long phrase . The motives usually occurred in a specific key, and over particular chord changes; however, the occurrences vary the archetypal motive through metric displacement, augmentation or diminution, addition and subtraction of notes, or altered phrasing and articulation. It is easy to understand Smith's criticism that there is nothing particularly motivic about these forms given this loose definition. For example, the use of four eighth notes in an ascending arpeggio is sufficiently basic as a musical option that it cannot seriously be considered to be distinctive motivically; such an appellation affords no new information or insight to the nature of the solos. The exhaustive statistical enumerations done by Owens provide a wealth of information about the frequency of superficial patterns, but offer very little insight as to how these patterns work, why they are present, or how they are put together into a successful solo.

Smith attempts to go one step beyond these lists of surface features by describing the solos of Bill Evans as examples of formulaic composition. A "formula" (in Smith's discussion) is the underlying pattern behind expressions that are similar in grammatical status or meaning, an idea inherited from the tradition of oral composition of poetry (which Smith is comparing jazz improvisation to). His analyses still focus upon simple superficial elements of the music, however, such as the direction and size of the steps in a melody; he emphasizes the similar metric placement of groups of notes as being a particularly formulaic characteristic. Part of the difficulty which he then encounters in listing formulae is that he allows himself only four categories: identical in form and substance; nearly identical; identical in form not substance; similar in form but not identical in anything. Further, his metrics of similarity almost completely ignore rhythmic concerns, and are inflexible in not being able to handle even the types of variation which Owens suggested. (The description of my system below avoids these pitfalls by not focusing on the misleading approach to similarity as a binary feature, instead emphasizing that similarity is an exhaustive listing of differences of properties, some of which are important for different functions.) Further, terms such as "melodic diction" and "discursive logic" which Smith uses to describe the presence of the formulae in the solos are not explicitly defined, instead referring to an impressionistic feeling about a composition. By better quantifying the basic elements behind his formulas, the combinations of these can then be explored as actual definitions of qualitative terms.

The overall problem with these reports is that they focus too much on surface analyses of music and unquantified similarity metrics. Indeed, repetition is abundant within and between solos, but the nature of this repetition is unsuccessfully grasped by simple syntactic rules, and the cognitive motivations for it are completely lacking. To say that jazz improvisation consists of inserting memorized fragments into a line at the appropriate time is to avoid the hard problem, which is to create a definition of how "appropriate" is determined in terms of musical processes.

2.3.2. Computer Programs

Many computer systems which attempt to emulate some aspect of human musical behavior have been created and are briefly summarized here. The positive trend in these systems is to aim for the performance of a specialized function through the encoding of a series of rules (frequently taken from music theory) which constrain its execution. My approach takes some elements from the successes of these systems, but is further motivated by their shortcomings: their inflexibility and absence of representation for common sense ideas.

A number of computer jazz systems have already been constructed, revealing a wide range of design approaches. One class of systems consists of relatively free-form arrangements for the computer's behavior, wherein the produced music is intended to be more an idiosyncratic display of the program than a representative imitation of a known genre. For example, Lewis [Lewis 94] has built a system designed to have a personality, that is, a distinct type of musical behavior governed by highly probabilistic choices from among harmonic and rhythmic elements of style hard-coded in the system (durations and scales are the main accumulated stock of knowledge). His system observes input music and calculates a few features from it, which are then available for the output generation routines to access occasionally, although the output frequently just "does its own thing." While these sorts of systems are interesting as artistic statements and explorations of new genres, they do little to directly aid in the understanding or production of any popular genres with which we are familiar; my interest here is in explicitly representing musical intelligence about well-known genres. Therefore, these systems are not discussed further here.

Cybernetic composer [Ames 92] is a generative system which concentrates upon producing well-known genres, but does so exclusively through the use of saved surface material. This program works by database lookup for fragmentary substitutions from a few standard rhythmic patterns and progressions typical of the chosen genre. These swaps are guided by templates, as is the mechanism included for following the schemes by which solos evolve (these indicate the imitative strictness over time). Surface information about jazz, from Mehegan's account [Mehegan 59], is literally encoded; pitch choice, for example, is constrained by instrumental range, melodic function (a role cleverly specified in the rhythmic description), a rule to resolve dissonances, and statistical feedback to avoid dull repetition (backtracking is also used to accommodate difficult situations such as choosing ornaments). This system is successful at generating artifacts bearing a superficial similarity to the modeled genre, for it literally uses fragments from these genres to construct a musical surface. However, it is limited by the absence of representations of the concepts motivating these surface conventions: for example, the different layers (instruments) in its generated textures are largely unaware of each other, for there is no representational currency with which they could communicate.

[Pennycook 93] offers a somewhat more flexible architectural approach, although still based upon the assumption that improvisational output is limited to a number of learned patterns, which are reducible to a few basic shapes. The "listener" in this interactive system does a segmentation of input phrases (using techniques from GTTM), extracts chord progression information, and constructs a "feature vector" of the input (representing density, register, speed, duration, and dynamic and harmonic function of events, where events are notes within 100ms). There is also a recognizer which tries to find similar patterns using the "dynamic timewarp algorithm"; the search for related contours is thus treated as a pattern recognition task, where the measure of similarity is the best path according to a Euclidean distance metric when the original and goal pattern are overlaid on a grid. While this successfully finds patterns which are mathematically similar, there is no consideration for what parts of the contour or other aspects of context are perceptually salient and could thus define a richer similarity metric.

[Levitt 81] presents an early model for a jazz improvisation system which is more inspired by an intention to model the cognitive and psychological factors of improvisation. The actual implementation mentioned in his document is simple and limited; it generates an improvisation by looking for a small number of surface features in arbitrary chunks of a melody (phrases are blindly broken at two measure boundaries, ignoring the line itself and its relative sense of closure), and then concatenating variations (each of which is appropriate for the found features) of these chunks. While this obviously does not address questions of the musical effects of these compound decisions (for example, when two fragments are concatenated they create a discontinuity which the program does not notice) or the multiplicity of perspectives which need to be acknowledged, his discussion of the approach towards modeling improvisation lays down some important initial ideas concerning the validity of examining improvisation in terms of the processes which create it. His later work on musical style [Levitt 85] is less useful on this front, for it emphasizes syntactic sets and templates as constraints to describe styles. A rough version of this approach is taken to its logical conclusion in the statistical analyses of [Cope 91], which manage to produce passably similar instances of certain composers' textures and basic forms given a set of constraints, but do not yield any information about the nature of musical understanding.

[Widmer 92, 94] stands out for his explicit acknowledgment that musical systems must be knowledge-intensive to function in a sophisticated manner reflecting the music's context. His systems are designed to learn rules, such as syntax rules or expressive performance rules, by taking advantage of the known structure of the domain (which is classical music). Although his systems are restrained by the simplicity of his choice of processes and representations, his postulations regarding an event/effect conception of music and his use of knowledge rules supported by multiple "determinations" are valuable.

[Riecken 92] presents a sonata-generation system which is a theoretical leap ahead from previous systems, largely due to his intention to take advantage of ideas from SOM in designing his architecture. He decomposes a sonata composition task into smaller elements, each governed by their own blackboard system. K-lines [Minsky 80] are used to enable memory of patterns, methods, and combinations thereof. An interesting step made by this program is to use "emotional" information to guide the processing; this is largely implemented by using simple evaluative functions which determine the appropriateness of a K-line's referent according to its emotional content compared with the desired content given the local context in the skeletal framework of the piece (which is determined by hard-coded sonata structure, a token element representing musical culture). "Good ideas" according to his evaluation criterion are given special status in the program, and thus become more likely candidates for later inclusion in the piece. This system is a rough initial attempt, noteworthy for the ideas it begins to implement, but also significantly incomplete in its actual incarnation (for instance, emotions are represented as one of four options applying solely to intervals -- a drastic oversimplification, as is the disregard for the effects of concatenation and combination of items, whose "emotive" values the program simply averages). This is a good example of how an implementation serves to reveal the explanatory gaps in a theory, such that it becomes clear that a quantification of the intended ideas requires further study.

There are several programs which are agent-based that should be mentioned here. The common use of agents is simply to separate processing for separate tasks, in particular the recognition or generation of distinct features [Hiraga 93; Rosenthal 92]. Hiraga uses networks of agents in a simple program to recognize melodies; his approach is noteworthy for its implementation of the idea that particular agents are generated and invoked by the relations they respond to, his listing of some ways in which agents interact (reinforcement and repulsion among similar types of agents, cooperation and suppression among different types, and competition between all types), and the obvious but often neglected consideration that a note's influence upon the current context extends beyond its actual time-span. His metrical agents are similar to Rosenthal's, recognizing segmentation ambiguity from the contrasting accounts suggested by different agents; he also notes that without grouping disagreements, music would be discontinuous. As is common with my own prior work with agents in music [Horowitz 94c], these systems do not have sufficient knowledge to extend their functionality, and are limited by the absence of other types of processing, such as the use of stronger rules or scripts or dependencies.

Cypher [Rowe 91] is a more well-rounded system than those just described, designed to allow for interactions over a wide range of styles. Input is represented as a vector in feature space, including information about how the features change over time. Two levels of agents are implemented, to indicate the separation between the tasks of identifying features of events and identifying features of groups of events, which take advantage of a message passing system. His system also attempts to maintain several perspectives on input material, using cross-connected processes through collections of ways to represent music. Output is computed by transformation of input along feature axes; the detailing of these transformations and mappings constitutes the "genre" settings for each performance. The output is determined solely through this method, rather than according to several simultaneous processes motivated to produce music over differing time scales, such that performing musicians have actually learned how to trigger specific transformations with their own playing. High level musical concepts are restricted to a few operations such as "vary density." Score cues are relied upon to handle some of the types of information not represented by the system, such as the conventions of how to "trade eights" with a live soloist. One of the best features of this system is its use of a "critic", an agency that can evaluate the system's output and allow it to have a sort of "introspection" through feedback and the accumulation of production rules for aesthetics. Cypher is a good example of a usable implementation of a few paradigms of representation and processing; it is mainly limited by its attempt at widespread practical functioning, which restricts the degree to which knowledge specific to a genre can be considered. As a result, a large realm of general musical intelligence which can only be explored through deeper study of musical genres is neglected; this is one explanation for the absence of other paradigms of processing and representation in this system. For example, its dependence upon a small fixed set of features (such as binary velocity) in a single time-window precludes representation of some basic elements of music structure (a demonstration of this difficulty is presented in the *Recognizing Similarity* section). The absence of representations of intention, abstraction, and context, also prevent this from being a complete expression of musical common sense understanding. However, while the solution to these problems is not yet known, it is at least approached through the incorporation of more musical concepts.

2.4. Intuitive Representations of Musical Knowledge

The previous sections have presented perspectives through which to consider musical understanding, first by discussing understanding and musical knowledge abstractly, and then by summarizing previous work which is relevant to this question. The current section presents a set of intuitive ideas about musical understanding which are motivated by the above sections; the ideas in the previous work have been generalized, and new ideas have been added to overcome limitations. The background for these ideas is the study of music, the study of processes which model intelligent abilities, and my own introspection as a musician and listener. These ideas constitute the theoretical approach which motivates the thesis system design: these ideas determine the elements of the design, and the functionality that the design must support in order to allow exploration of more complex ideas.

The discussion included in this section is thus the inspirational blueprint for the thesis system. This section is not intended to be a scientific description of music; rather, it is meant as an intuitive and abstract discussion which motivates the formalization and realization of specific ideas in the thesis system. The topics covered below are intended to convey as concisely as possible the intuition behind my approach to thinking about musical understanding, so as to explain the choices made in the system design.

2.4.1. Musical Concepts

Musical common sense ideas can be viewed as an interconnected network of concepts about music: the properties of musical phenomena in relation to each other and under transformations. Each of the properties can be a simple parameter value, or can itself be another type of concept. Every musical concept can appear in a variety of instantiations, varying in their degree of abstractness and the values of their parameters. Thus the concepts define a space of musical possibilities; a given point in the space is a set of parameter values for the musical concepts.

We would like to have an explicit representation for each of these concepts. A simple organization of (suggested by SOM) is to create an agent for each concept; the agent is a module which contains all the localized knowledge about the phenomenon. It is able to detect an occurrence of the concept it corresponds to, and is also able to effect an occurrence of its concept. It knows the properties of the phenomenon, it has a model of the space of possibilities permitted by the variations of these properties, and it has links to other agents that are related to it (agents which it requires information from or provides information to).

Common sense concepts describe the following basic features of a melodic line. The structuring of notes in a harmonic framework (such as a Western diatonic framework) includes knowledge of chords, scales, and their relations. The structuring of time in a metric framework provides a hierarchical arrangement of levels through which an ideal binary organization of time can be achieved. The features of individual notes must be represented, in terms of their intrinsic values of pitch, loudness, etc., and also in terms of their relation to a metric and harmonic frameworks. The movement between notes must also be categorized, so as to explain the nature of interval motion in relation to the harmonic framework, and of the movement through time in relation to the metric framework. By induction, the movement between notes defines the qualities of sequences of notes; these sequences are describable in terms of the contours of their features over time, which can be further examined as regions of grouped similar features (such that discontinuities and changes of feature values are evident in the contour). Sequences of notes abstracted from the musical surface must also be possible, to reflect interpretations of skeletal structures of a line. These sequences are segmented into phrases, which have generalized features for their constituent notes and abstract lines. Sequences of phrases can then be described in terms of the relations of the qualities of successive phrases. The sum of these elements constitutes a general simplified set of characteristics about melodic lines articulated by music theory and common sense understanding.

These basic features thus provide the groundwork for slightly more complicated features. An appoggiatura can be defined in terms of pairs of notes, a feeling of forward motion can be explained by the occurrence of a note in an anticipatory position in relation to the metric framework, and an abrupt syncopated ending of a phrase can be defined in terms of the properties of a note sequence. These features can also indicate conventions in the music: the discontinuity of a leap in a melodic line, and the perceived logic of an ensuing stepwise motion in the opposite direction, are instances of a standard note sequence; the conventions for how to structure a solo over the chord changes near the end of a chorus can be defined in terms of the properties of note sequences in relation to metric and harmonic structures.

We can define a genre of music in terms of its conventions for the properties of these concepts in relation to each other. For instance, the usual placement of certain pitches in succession to define a major scale characterizes conventional melodic motion of much Western music, while the position and relations of phrases within a song's chorus characterize jazz solos. Each genre then determines the affective quality of a given phenomenon in terms of its intrinsic color, combined with its degree of submission to the conventions for what is expected. (As we saw with the implication-realization model, instances of music can have affective meaning defined by the expectations which they create in relation to their genre.) The most common basic expected features, such as the resolution of unstable tones or metric positions, take on the status of being sub-conscious urges in an experienced practitioner of the genre. Different levels of concepts can have conflicting expectations, such that a low level can expect a resolution, even though a higher level defines the absence of such resolutions within the genre.

2.4.2. Similarity and Labels

Music is distinguished by this sublimely vulgar excess of redundancy.
- Minsky: [Minsky and Laske 92]

In addition to the processing effects of music structures, such as the affective theory of implication-realization, musical meaning derives from the references of instances of music to each other. This is evident in our evaluation of music: tunes are generally more acceptable when they are familiar, a song is more easily comprehended when its sections refer to each other, etc. This important concept of reference, however, is only one manifestation of a more basic concept of musical similarity. Repetitiveness and self-similarity abound in music, both within and between musical compositions.

Minsky suggests that one of the purposes of this extraordinary repetitiveness within a piece of music is to allow for the perceived emphasis of slight differences. For example, the use of a metric framework provides a simple indication of how to align consecutive phrases for comparison. Once they are aligned, a listener might notice that the phrases are the same except for the inflection of the last note of the second phrase. This comparison then becomes a stored representation of the music, focusing upon a salient feature. This is only one example of how representations are formed, and how comparisons are made. Yet even this small example presumes a large amount of common sense knowledge. The metric framework has to be understood, and several slight shifts have to be made to ensure the alignment of the phrases. More importantly, metrics of similarity themselves have to be understood, such that transpositions, inversions, diminutions, elaborations, simpflications, etc. are all recognized as transformations from the first phrase's represented features into the second's; otherwise, the second phrase appears as a completely new list of arbitrary notes.

To describe the similarity between pieces of music is a complex problem, and can only be satisfactorily approached by preparing a large set of mechanisms for evaluating types of similarity between particular features, and then determining which of these are appropriate and important for the specific situations. Initially, we need at least to be able to have each agent for each basic musical feature be able to compare instances of its concept by comparing their parameter values; where these values are themselves concepts, the comparison is applied recursively. There are an infinite number of predicates possible for any object, so the important aspect of this enumeration is to attempt to find a set of simple predicates which are combinable to cover a large range of these possibilities. Part of musical common sense knowledge is knowing what the important features are for musical comparisons by default. The important structural properties of a phrase are the likely ones by which to determine similarity in the absence of other features. However, if two structurally similar phrases are played over different harmonies, this implicitly indicates that it is their relation to the harmony that is relevant. The mechanisms of music contain these types of clues to direct the listener's attention to what is important. By repeating some elements and changing others, the musician directs attention to the differences.

For example, the opening two bars of "Heebie Jeebies" (see *Example Comparison #1* section) are easily comprehended by human listeners as a statement and a varied repetition of a phrase. This comprehension results from the properties of the phrases themselves, and from the knowledge that it is conventional in this genre to have repetitions at similar positions with respect to a metric hierarchy. The first interpretation of this music is the natural comparison between these two bars: it indicates that the second phrase is shifted one beat later than the first in relation to the measure, that it adds an eighth note elaboration between the two D's, and that the final F occurs early in relation to the rest of the figure, anticipating its expected position. However, a second interpretation is objectively feasible: to compare these bars by stating that the second phrase deletes the first D, adds the eighth as described above, and then moves the first phrase's F to a D, and then adds a new F to the end of the sequence. This latter interpretation is much less intuitively appealing to most listeners; it does not correspond to our common sense analysis of what has occurred. In order to reflect this, a system must be able to notice the importance of the similarity of the skeletal pitch contour (the first interpretation), to determine that this is more salient than the presence of different pitches at particular metric slots (the second interpretation) in constituting the match from which a comparison of features can be made (by overlaying the specifics of each instance relative to the position of the assumed match).

2.4.3. Transformations and Variations

The creation and comprehension of variations, or of the development and evolution of a musical idea, requires comparing fragments of music and determining what transformations relate them. These relations can then themselves be categorized to describe the structures of entire solos, or of nuclear phrase forms.

Discourse structure and narrative structure are likely sources of inspiration for models of the development of music over time; the discussion below focus on the musical terms necessary to support such analogies. This section summarizes the possible transformations on the basic features of music that are important in determining similarity.

Every concept can transform one of its instances by changing its parameter values. For example, a pitch can be transposed to another pitch, a stepwise motion can be turned into an arpeggiation, or a sequence can be inverted into a set of intervals with opposite directions. The compound effects of these are interesting, for they constitute transformations of actual phrases. The following two paragraphs enumerate important phrase transformations which can be made by changing rhythmic or melodic features.

Rhythmically, the character of a phrase can be varied by adding or deleting notes at the lower regions of the metric hierarchy (changing its perceived activity), by shifting notes or emphases to off-beat positions (changing its perceived syncopation), or by adding ametric emphases (changing its perceived crossrhythmic feeling)[12]. The number of notes in a figure, its overall length, or its placement in the meter can be maintained (while varying the other features) to create a rhythm that is substitutable for the original. In addition, the presence of different levels of salient cycles (strings of emphasized notes separated by an even period) determines the character of the figure, as does the lowest played subdivision (the smallest pulse); changing the pulse level or adding/removing salient cycles transforms the pattern, as does shifting an entire cycle relative to meter. Finally, the character of the first and last notes of the figure can be shifted to align with the meter, to create a smooth start/ending, or to misalign, to create and abrupt and unexpected start/ending.

Melodically, a pitch sequence can be altered by transpositions, inversions, and stretches. Failing to start or end the phrase on resolved pitches, or emphasizing unstable pitches, provides a distinct color to a phrase. The contour of intervals and changes of direction is a salient abstraction of a pitch line. Maintaining important pitches in the contour, those that serve as the skeleton or anchor of the phrase (such as contour peaks), while changing others results in a new phrase with a similar overall melodic function. An extreme example of this is keeping the first and last important pitches of a phrase while changing the others; by viewing the phrase as an elaboration of its basic progression from start to end, properties such as its perceived motion and directness can be modeled. Elaborations can lead to a goal tone as progressions, or follow an important tone as prolongations (such as repetitions). The perceived melodic range of a phrase is determined by the combination of the vertical (registral) distance covered and the harmonic (chord degree) distance covered; in addition to the overall range, the melodic motion of a phrase can sum each of the individual steps to determine the total ground covered, which affects the perceived activity. The type of scale used in each portion of the sequence directly determines its color; it is often possible to maintain the same skeletal contour while changing type of scale. Finally, by shifting pitches with respect to accented metric positions and other emphases, new alignments are created which result in distinct melodic qualities, such as apoggiaturas.

These types of structural transformations can be combined and categorized to create qualitative effects and high level changes, such as the changing the affective quality of a phrase. While affective qualities are somewhat subjective, the important point here is to be able to categorize similar types of affects, regardless of how they are labeled. Thus, each of the transformations above describes a relation that represents an affective difference; in the context of a system without extra-musical references, the best possible label of an affective character is simply a collection of the concepts and parameter values which reflect the character. Entire gestures can then be categorized in terms of their similar features.

3. Building a More Intelligent Music System

This section describes the system which I have designed as an example of my approach to building more intelligent music systems. This design is an attempt to deal with the limitations of previous representations of music, and thus is intended to be incrementally better than previous systems while also being extendible to address further questions of musical intelligence. It is inspired by the intuitive representations of music discussed above, which have been quantified into the constructs below as representations of common sense concepts about music. The design decisions were made with the goal of creating a flexible framework with which the ideas in the *Understanding Music* section can be explored; this design literally implements some of these ideas, and can be extended to handle others.

The design of the system consisted of enumerating the domain knowledge necessary for intelligent behavior, creating the representations to contain this, and building processes that could actually perform the behavior. The *Problem Specification* section describes the specific domain and task towards which the system's design is oriented. The *System Components* section describes the overall layout of the system, including its SOM-inspired architecture. The *Representing Music* section describes concepts that are present in the system, motivated by the *Intuitive Representations of Musical Knowledge* section. The *Processing Music* section describes how these representations are used; this explains the details of how notes entering the system are managed, and how the system generates output notes.

3.1. Problem Specification

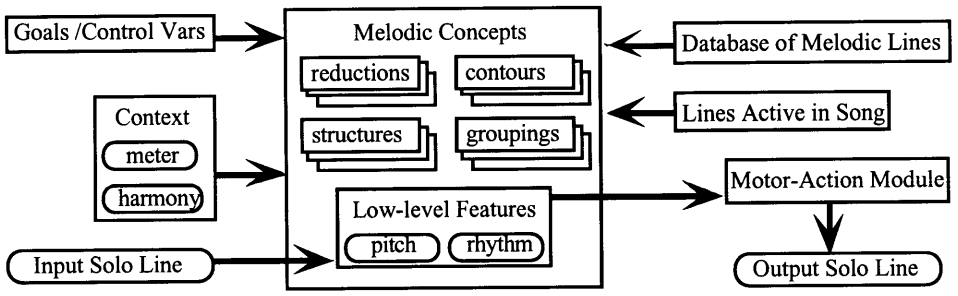
The system has been designed to handle a particular domain and task in order to provide a concrete example of my approach. The domain and task were chosen because of their natural emphasis on the abilities of comparing music and generating variations, which are representative of the type of common sense musical abilities which we would like for a program to have. As these abilities are common to many types of music, much of this system design is generalizable to other domains and tasks as well.

The model presented in this section focuses on the style of music performed by Louis Armstrong in his famous recordings with the Hot Fives in 1926. The studied songs on these recordings are usually in 12-bar blues and 32-bar song formats, where the trumpet solos are monophonic lines played on top of the backdrop of a rhythm section. This sub-genre has been chosen because it represents a foundational moment in the history of jazz, one that must be understood in order to understand subsequent developments [Schuller 68]. Armstrong had come from a background with King Oliver's band, which played in the New Orleans style known for collective improvisation, consisting largely of arpeggiation on the current chord and elaboration of the song's main tune. With the Hot Fives, Armstrong initiated the more modern concept of improvisation as individual free melodic invention [Collier 83].

The appeal of this domain for an example system is twofold. First, the question of whether or not a system is musically intelligent in the sense that humans are is best pursued in a familiar genre where its performance can be judged and compared to human performance; all too frequently, the opposite approach is taken and the curious behavior of a system which is called intelligent is cloaked behind the mystique of an unfamiliar new genre. Second, improvisation as a generative behavior appears to be more susceptible to modeling than non-real-time composition. Although most improvisers use previously developed material in their solos, there is still a process by which decisions are being made on a moment-to-moment basis and are reflected immediately in the output music. We would like to build a system that can provide a model of this process, such that we can change its parameters and immediately observe a change in behavior. Minsky provides another perspective on this idea, suggesting that improvisers cannot cover their processing tracks easily, so they might be able to communicate better; that is, their processes are more evident and thus more easily perceived by the listener, which constitutes communication [Minsky and Laske 92].

The main task for the program based upon the model is to analyze and generate improvisations in 12-bar blues and 32-bar song formats which are similar to a target set of actual Armstrong solos from this period[13]. With this task in mind, the goal of the model is to functionally describe the musical style in terms of its important structures and the processes that parse and produce it. The model proposed here provides a knowledge representation and control structure which can perform these analysis and generation tasks. Thus, this model serves as a working theory of improvisational behavior within this sub-genre.

3.2. System Components



3.2.1. Logistics/Utilities

The physical components of the system are a keyboard MIDI input device, a PowerMac 8100 computer with MIDI interface, and a MIDI synthesizer; these devices are chosen for convenience in building the software, and can be replaced by other configurations of I/O devices for particular applications. The system operates in real-time; it receives an input MIDI stream from the keyboard, and generates an output MIDI stream to the synthesizer. The system is written on top of a generic scheduler I wrote using Apple's MIDI Manager. Running concurrently to the main system is a simple *demon* program that plays the role of the rest of the band, providing the accompaniment to the soloist. Given a chorus structure, the demon generates a simple sequence-like bass line, drum part, and chord accompaniment part. The main system then polls the demon to determine the position of an event's time relative to the beat and the chorus[14].

3.2.2. Architecture/Organization

The architecture of the system is inspired by the model of mind presented in SOM. This architecture allows the system to represent a piece of music in terms of the different musical concepts that it relates to. As explained below, each of the slots (terminals) in a concept-frame is a perspective on the music which is useful for a specific purpose. The important aspect of the SOM model used here is this separation of the representation and functionality into specialized features, so that an object can be considered in terms of the appropriate perspective for each of the functions that uses it[15].

Agents and Frames

Musical concepts are managed by specialized *agents*[16]. Each type of concept has an agent that describes it and is able to perform calculations relating to it. The agent manages the *components* of the concept; it lists their relations, and the *parameters* which determine the range of different instantiations of the concept. The agent also knows the *context* in which different instances occur (analogous to *preconditions* for each possible instance's occurrence), and the characteristic *color* effects of their occurence (analogous to *postconditions* of its occurrence)[17]. Finally, the agent includes pointers to other agents for which its concept plays a role. By separating concepts into agents, each agent can be an expert in a single specialized and simple task; it need only be aware of that information which relates to its concept, thus insulating it from the complexity of dealing with unnecessary information. Each agent is able to detect the occurrence of the concept that it corresponds to through *sensors*, and also to effect an occurrence of an instance of the concept through *actuators*. Following the object-oriented paradigm[18], there is a top level agent class, which provides generic agent functionality, from which all other agent subclasses inherit; not all of the functionality is used by each subclass, and each subclass adds its own specializations of the generic model.

Each type of agent has a corresponding *frame* [19] that contains the slots (variables for holding the parameters) which are filled in by particular values for each instance of the agent's concept. Each of the frame slots can be either a number or a pointer to any kind of data structure, including another frame. In other words, a frame consists of the recorded state values of the agent; an actual object (such as a particular phrase) in the system is an instantiation of a frame class. Each subsection in *Representing Music* corresponds to a type of frame in the system. By convention, the names of frames and slots are capitalized to indicate that they are symbols in the system, and the agent corresponding to a frame class is referred to by the name of the frame. Following the SOM model described above, the calculations required to fill in the slots of the frames are handled by their corresponding agents. Minsky suggests that we need only remember fragments of an instance, for many details of its occurrence can be filled in by default. In this system, this is literally implemented by having default values for frames which are used when no specific contrary information is available.

Network of Agents and Instances

Thus, the system's knowledge of music consists of sets of agents which represent musical concepts. These are arranged in an interconnected network which contains links for the different hierarchical arrangements of the agents, reflecting their relations: each agent has links to the agents which fill its slots, and to the agents whose slots it fills; each agent has links to the agents which determine its context (satisfy its preconditions) or use its effects (require its postconditions). The details of these represented musical concepts are described in the *Representing Music* section. These are then used by processes for listening and generation, as described in the *Processing Music* section. (Most of these processes are handled by the agents which define the representation. In addition, the generation processes include an extra utility MotorModule for handling logistics of note output not directly related to the representation.)

The system also contains a database of known material, which includes conventional song/chorus forms (described in *SoloFrames* section) and instances of melodic lines (well-known figures, scale conventions, or tune fragments). The instances of melodic lines are saved instances of frames in the representational scheme described below. In addition to the known material, the system maintains a temporary buffer which holds saved instances of melodic lines played previously in the song/session of the program's execution. Each saved instance has an Activation value (see *Processing Music* section) which reflects its temporary eligibility to be considered for matching (in listening) or reference (when playing). The instances are stored in lists, but are also related through links: saved instances are linked according to their sequential relation, so that they can trigger each other to become active; each agent's instances are grouped separately, so that the agent can bring its particular instances into consideration.

Interface and Control

The utilities, the agents, the saved instances, and a console interface are connected in a single global class that manages the control of the system. It contains a set of *control variables* that serve as parameters to the operation of the system; these are the "hooks" for the interface, numbers that are used to change or reflect the way that the system behaves. They include coefficients for biases that the system has in evaluating input (such as thresholds), and numbers that reflect its disposition in making choices for generation. By changing these numbers the system can be made to behave differently; the specific uses for these control variables are described in the sections below. The qualitative aspects of the system's generated solos are controlled through a *UserSolo* structure in this class, described in the *SoloFrames* section; this contains the specific representation data that passes between the user and the program.

The graphic interface (currently under development; see *Appendix 1*) will allow users to investigate the workings of the model by adjusting control variables and the UserSolo structure in the system; currently, interaction with the program is limited to commands in console window. This is the most simple mode of interaction with the program, allowing direct evaluation of the theory behind it. However, there are a wide variety of possible demonstrations/applications for systems with increased musical understanding; for example, an interactive system that can participate in the jazz convention of "trading fours" (in which soloists alternate in playing short related improvisational phrases) demonstrates both listening and generative abilities.

3.3. Representing Music

This thesis presents a representation of music through the design of a symbolic system that can analyze and generate music. The sections below present the representation of music that this system contains for its specific functions, and simultaneously explain the architecture in which this representation is encoded. This section and the *Processing Music* section separate the elements of the system into knowledge of structures and knowledge of processes, but this distinction is somewhat artificial and is adopted here for explanatory convenience; it is often blurred, such that the current section discusses elements of the processes which use the representations. As the descriptions will show, the nature of a process itself sometimes constitutes the system's knowledge of a structure, and a simple traversal of a data structure is sometimes the definition of a process.

3.3.1. Representation and Knowledge Criteria

Ideally, the knowledge represented in the system should be the set of structures necessary to form the conceptions of solos which are articulated by musicians, music theorists, and listeners. This thesis assumes that this knowledge constitutes the terms that are used for musical reference within the solos themselves[20].

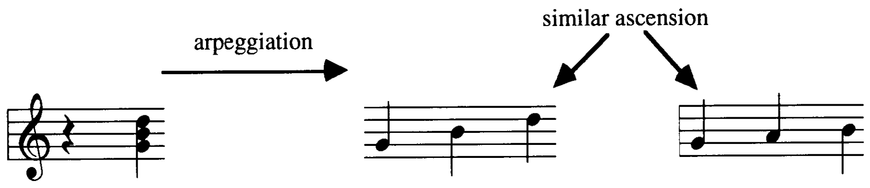
The first step in the design of a system is the determination of what knowledge is necessary to explain the material in the domain. In other words, we must determine what concepts and categories are recognized as salient by humans in the material, and also what background knowledge informs this recognition process. Once the types of knowledge that contribute to human understanding have been enumerated, the next step is to create a representational structure which can contain this knowledge. The main criteria for the representation are that it reflects clearly and explicitly the information that it contains, and that it affords manipulation of the information according to appropriate constraints.

Beyond simply recording input data, listening and understanding involve categorizing input. We claim that an instance of music has been analyzed when we have labeled it according to the categories that it belongs to, noted its specific parameters, and placed it in a network which expresses its relationships and similarities to other instances of music and to general concepts and types of manipulations. This network constitutes the system's knowledge of music; the interconnected placement of an instance in the network constitutes an understanding of the instance. The generalizations of instances are categories, which are the frames of our system; the presence of frames as categories enables the reception of new material of that type. Thus, a particular melodic line can be seen as a specific instantiation of general abstract types of concepts. This abstraction allows for comparison of musical figures according to their general types, and provides a rich set of metrics for determining their similarity or relation; this information is necessary for understanding music, and for generating variations.

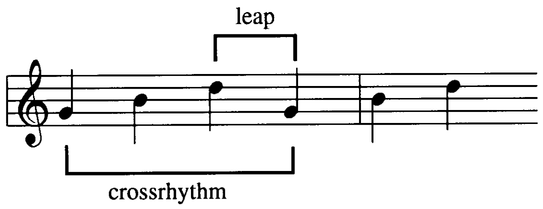
Recognizing Similarity

A representation of music should reflect the types of operations or transformations that are possible on the music. Another perspective on this criteria is that the representation should define a similarity metric for instances of music. The representation is an explicit quantification and organization of features of music that are perceivable and salient to humans. Thus, a goal of the representation is to have similar instances of music represented similarly. While this may seem to be a trivial statement, it in fact speaks directly to the deficiencies of several previously used representations.

For example, a representation of music in which a chord is a set of notes which can be subjected to the operations of "arpeggiate" and "loop" (repeat, or sequence) will fail to represent the salient features which arise from these operations. In particular, the "arpeggiate" operation on a chord produces a line which is similar to a simple scale ascension:



Similarly, a "loop" of this arpeggiation may result in a leap or a cross rhythm that should be reflected in the representation:



My approach emphasizes the use of multiple perspectives, subsuming this example representation. The benefit of multiple representations is that a single musical line can be manipulated in different ways depending upon which aspects of it (i.e. which agents' features) are considered. A simple example of this is the separation of rhythmic and pitch information about a line; a more complex example is the separation of a prolongation reduction perspective from the set of local elaboration figures that are used. In the *Melodic Line Knowledge* section, I start from lists of timed notes to develop a set of higher level structures which seek to provide a more complete representation of a melodic line.

There are many ways in which phrases or fragments of phrases can be similar theoretically; those which are salient (noticeable to humans) are the ones which have become part of our representation here. As each part of the representation is presented, an example is shown which demonstrates the type of similarity which this representation admits (that is, how melodic fragments can be compared from the perspective of the representation). By the end of this discussion, the representation will be sufficiently outlined to permit comparisons of entire phrases. The use of multiple representations of a single piece of music is the basis of this design, and is necessary to develop a system capable of the complexities of human musical processing.

The following sections describe the knowledge which I propose a system must possess in order to produce analytic and generative behavior demonstrative of an understanding of the solos of Louis Armstrong in the late 1920's. Many aspects of this knowledge are applicable to other genres of music as well, and the general framework for organizing the information is extensible to handle further additions. In keeping with my approach, the important aspect of this discussion is the explicit enumeration of the types of knowledge present in the system. I have divided the knowledge into two categories: 1) background knowledge, describing the program's understanding of the form of a song, and of time and harmony; 2) knowledge about a melodic line[21]. The bulk of the design work has been in the development of a representation of melodic line, one that will support an understanding of the conventions which govern a solo's development over the course of a chorus in a song form.

3.3.2. Background Knowledge

Song Knowledge

A single Hot Fives' song is a combination of a melodic line (the *tune*) and a sequence of chord changes which harmonize the tune. The tune and the chord changes usually follow a simple form of repeating sections such as AA'BA or ABAC (where each letter represents a distinct section), a single traversal of which is called a *chorus*. A performance of a song usually begins with one chorus in which the tune is played, followed by several choruses of the chord changes over which the band members play solos, and then a final chorus which repeats the tune. Occasionally this form is varied (e.g., an introductory solo break or closing cadenza is added), but these variations are not important for our purposes. This description of jazz song form is well documented in several sources [Schuller 68].

A computer system aimed towards understanding solos must have a representation of the underlying song over which the solo is played. The simplest representation is a list of chords indexed by their placement on specific beats in measures within a chorus. This representation constitutes an implicit set of assertions about how time is structured through meter, and how harmony defines and relates chords.

Time and Meter

Traditional music theory describes musical time as consisting of hierarchically arranged units of evenly spaced pulses, each level of which is a *cycle*; two adjacent pulses in a cycle are a *MetricStep* (or a *CycleStep* if the cycle is ametric, not part of the metric hierarchy). The periodicity of each level of pulse seems to be of fundamental importance to music's structure. [Lerdahl and Jackendoff 83] provide a traditional example of this type of metrical hierarchy. Each level is said to consist of evenly spaced events, where a pair or triad of events on each level is reduced to a single event on the higher level. This description of meter is intended to correspond to musicians' intuitions about how larger sections of time are organized, as well as to suggest a model of the perceived iambic or alternating accent/emphasis on each level. A discussion of foundational music perception experiments attempting to verify this theoretical description can be found in *Appendix 2*.

Following this theory, the system's representation of time consists of a MetricForm agent which is capable of forming groups out of pairs of events, and applying this operation recursively. The frame which corresponds to the meter of a song is represented as an instance of this agent's state, with groups organized in the ideally uniform binary tree structure for a 4/4 32-bar composition, or with slight variations to allow three nodes instead of two on some levels in those song forms which require this. Each node in this hierarchy knows about its position in the hierarchy and its relation to the other nodes, both in terms of its absolute position on the surface level and its relative position within higher level groups. The benefit of Minsky's use of frames to fill in default terminals serves here to avoid having to explicitly figure out all of the relationships between metrical elements; only a little information needs to be specified for each new instance, and the default connections are made.

The chorus of the song is described as one high-level unit of this metrical structure in which the chord changes are placed. The entire song form is then described as an optional introduction or closing cadenza on either side of a pair of choruses in which the tune is played, and which are separated by a variable number of choruses in which solos are played. The simple data structure that holds this information thus suffices for being able to indicate the "meaning" of any time position within the song: for example, an event may occur during the second solo, in the first bar, which has a current harmony, and which is a specific section of the tune's form. This structure is used as a utility to determine the necessary background song context for understanding the melodic line.

The standard cycles for the levels in a metric hierarchy of a chorus are given the following names in the system: EighthCycle, BeatCycle, HalfCycle, WholeCycle, TwoMeasCycle, FourMeasCycle, EightMeasCycle, SixTMeasCycle, ChorusCycle. Ametric cycles are given the common name of CrossCycle, but may have different periods (although the dominant period for a cross rhythm in this genre is a dotted quarter, appearing in varying phases). The metric cycles are part of the learned conventions of the music; by default, they are expected by listeners, and felt by musicians as fundamental default urges. If each level contributes an equal amount to the emphasis of a given slot on the lowest level, the result is a natural sounding iambic accentuation pattern; distortions of this natural order, either by shifted emphases or missing slots, constitute syncopations[22].

While the idea of listing chord changes in some representation of a chorus' meter is not unique to this system, using a hierarchically arranged set of nodes each of which has the primitive ability to determine information about its surroundings provides the system with a richer context than a simple list representation can. For example, any comparison of positions can indicate on which levels of the hierarchy they are the same or different in terms of being on- or off- beat on each level; this information is used in by the listening and generative processes to activate likely matches between current material and memorized figures. Further, this structure has pointers on each level which constitute the links over which upbeats and afterbeats progress. The MetricForm agent which produces these nodes as its saved state instances is able to find ametric periodic cycles as well, and share the knowledge of metric structure with these. This is described later in the *Melodic Line Knowledge* section.

Harmony, Chords, and Scales

The usual representation of a chord or a scale is simply a set of pitches. However, just as is the case with meter, the hierarchical structure in which chords and pitches relate must be made explicit in order to understand the meaning of any pitch harmonically. A few fundamentals of harmony are presented here in the course of describing the system's representation of harmony. These definitions here are not intended to be (and in fact are not) either revolutionary or universal as music theoretic statements; rather, they are an elementary example of the type of harmonic information which can be present in the system, chosen for their simplicity and fundamental relevance to the genre.

Each of the beats in the metrical form of a chorus has a current chord label (a default label of -1 indicates that the current chord is inherited from the previous beat). The chord consists of a root tone (an integer from 0 to 11, such as F[=5] or C[=0], termed the "tonic") and a type (an enum token, such as MAJ7, DIM), which determines the set of tones layered at intervals (usually of a third) above the tonic. (The member tones of the relevant chord types are listed in the PitchSeries table below.) The inversions of a chord refer to voicings in which a non-root tone is placed on the bottom of the chord. Traditional music theory describes the harmonic distance between two chords as being the number of steps of a fifth that separate their roots (C to G is 1, C to D is 2). From this information, the system can calculate the harmonic distance from the key of the tune that a given chord is, as well as the harmonic distance between successive chords; this integer label categorizes the type of progression. Each type of progression has associated characteristics: currently, the only characteristic represented is the set of known instances of melodic lines played over the progression (e.g., figures played over a turnaround); extensions of the design could add the associated affective color of the type of chord progression, other material which occurs in relation to it, and types of properties likely to be found (e.g., chromaticisms including the successive chord's leading tones are introduced near chord changes) [Bharucha 94].

Every pitch (in MIDI, an integer from 0 to 127) can be represented relative to a chord as an octave register (floor((pitch - chord_root) / 12)) and a degree ((pitch - chord_root) mod 12). Given the twelve pitch degrees, certain traversals of these tones are common in this genre. The most common are arpeggiations of the current chord (playing the notes of the chord sequentially), and major scale step motion; these are the default expected characteristics of pitch motion, much as the metric hierarchy is the default expectation of time structure. There are several other types of traversals that are common and yield a characteristic sound, including use of the other diatonic modes, octatonic and pentatonic scales, and arpeggiation of different chord types. The system refers to each of these types of traversal as a PitchSeries; a given PitchSeries is simply an ordered list of pitch degrees. (A PitchSeries is intuitively similar to a scale, but also includes non-scale types such as arpeggiation.) Given a PitchSeries, a pair of pitches is said to be an instance of the PitchSeries' stepwise motion if the pitches are adjacent in the PitchSeries' list[23]. In other words, an ascending major scale is represented as a linked succession of instances of steps in the MAJOR-SC PitchSeries. The set of PitchSeries types known by the system is easily extendible to handle new cases outside of those necessary to handle the studied set of solos. Here are the types of PitchSeries currently used by the system:

PitchSeries name:	Ordered List of Pitch Degrees:
MAJOR-SC	0, 2, 4, 5, 7, 9, 11
MINOR-SC	0, 2, 3, 5, 7, 8, 10
CHROM-SC	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
PENT-SC	0, 3, 5, 7, 10
MAJ7-ARP	0, 4, 7, 11
MIN7-ARP	0, 3, 7, 10
DOM7-ARP	0, 4, 7, 10
MAJ6-ARP	0, 4, 7, 9
DIM7-ARP	0, 3, 6, 9
TONIC-FIFTH	0, 7

In addition, each of the twelve pitch degrees has a Harmonic Tension label (an integer) that indicates how unstable the pitch is based upon the possibilities for inclusion in the above PitchSeries. Intuitively, being a member of several PitchSeries constitutes greater stability, for each PitchSeries offers an interpretation of the pitch's stability; whereas if a pitch is only in one PitchSeries, from the perspective of the others it needs to resolve to another pitch which is in their series in order to be explained functionally. The relative Harmonic Tension values shown below are the simple quantification of this metric based upon the current set of PitchSeries; as the set of PitchSeries used in the system change, the system's concept of relative Harmonic Tension also changes.

Pitch Degree:	0	1	2	3	4	5	6	7	8	9	10	11
Relative Harmonic Tension:	0	9	7	5	5	6	8	1	8	6	5	7

The combination of the set of PitchSeries with the PitchDegree's Harmonic Tension allows for a quantified definition of harmonic resolution in a melodic line: resolution occurs when there is a decrease in the harmonic tension between two successive notes which constitute a step in one of the PitchSeries (this excludes motion from degree 11 to degree 3, for instance). The amount of perceived resolution is in proportion to the integer change in Harmonic Tension.

Structure, Form, and Context

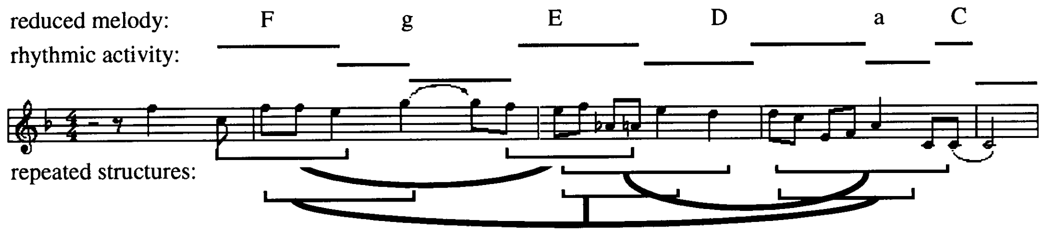
A large portion of Western music is organized using high-level structures (such as the metric hierarchy above) which contain discrete sections of music; these structures usually follow certain conventional forms. The structure indicates what the relationships are between the material in the sections: in particular, some sections are repetitions or variations of others. As we saw above, the performance of a song in a Hot Fives recording session is likely to follow a certain form for the order of the choruses. Within a single chorus of a tune or an improvisation, there is also a conventional structure; for example, AA'BA form refers to the repetition of the chord pattern (or tune) of the first eight bars (A) in the second eight bars with a slight variation (A'), followed by a new chord pattern for eight bars (B), and ending with a restatement of the first eight bar pattern. In a melodic line, a similar conventional pattern of repetition at increasingly higher levels of a binary measure hierarchy occurs, as described in the *PhraseRelationFrames* section.

Given these structural representations of time, harmony, and phrases themselves, musical context can be simply defined by position relative to each of these features. For example, the context of a specific riff in an input line reflects the harmony over which it occurred, the relation of this harmony to the adjacent harmonies, the position in the metric structure in which it occurred, and its relation to its preceding and following figures (structural repetition or variation in a phrase). These terms of context can then be listed as constraints in preconditions for the execution of saved instances of material that are registered with separate agents.

3.3.3. Melodic Line Knowledge

Intuitive Description

The simplest representation of a melodic line is a sequence of timed notes. However, there are properties of a melodic line that listeners and musicians notice: some notes and features are more important than others in different contexts and from different perspectives and biases. A line has basic properties determined by its surface features: the progression of tones (along a PitchSeries or otherwise) and the series of rhythmic slots that it fills, and the harmonic and temporal ground covered by these. These surface features provide a qualitative color defined by their continuity and direction, and the instances that they relate to. In addition, abstract skeletal structures are implied by the features of the surface; each of these skeletal structures itself constitutes a line, with its own color and rhythm. The various interpretations of these structures are the characteristic ambiguity of the line. The diagram below shows a melodic line from three perspectives represented in the model.



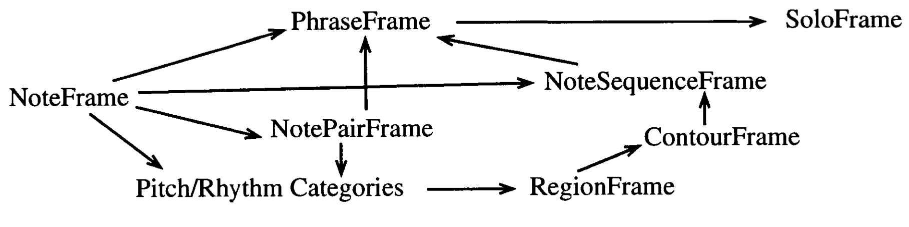
Melodic lines are heard and produced in an environment of learned conventions and expectations. The "semantics" of a melodic line is its relationship to conventions, previous material, and to the underlying meter and harmony; these relations define the conventions and expectations within a genre. The color of a line is affected by the sequence of alignments and misalignments of its tones and rhythms with the stable structures presented by the chorus form, and by the relation of its structures to the other lines which it refers to. The relations to metric and harmonic structure determine the line's degree of closure and resolution; the relations to previous lines reflect interpretations or reconstructions of those lines.

Every note indicates the role it plays in the different levels of the containing phrase's skeleton, which in turn determines whether that note is useful for a given function. For example, we might say that the first note of a phrase is the accessor through which we start replaying the phrase in our imagination, but it is really the first foundational tone that we remember as being what the phrase is about in a reduction, while we also focus on the particular elaborating tone that gave the phrase its distinctive quality. Maintaining these different perspectives is useful for both generating and analyzing material. These concepts serve as filters through which we can examine a phrase in order to determine its qualitative effects and its similarity to other phrases from different perspectives. Identifying these levels in the musical surface allows us to have a notion of a phrase that is richly interwoven with our general knowledge about types of phrases. It is through these devices that the concept of one phrase referring to another is realized.

In order to understand a melodic line, we must understand these elements. We must know what about a line is distinctive, and what is a reference to a previous phrase or a conventional motion. To capture this nature of a melodic line, its rich perceived structure and character, the following representation explicitly represents these elements, thus emphasizing the use of multiple perspectives of a phrase. Note that the intention here is not for the system to decide upon a single reduction of a line as its analysis;. Rather, the exhaustive collection of possible interpretations produced by human cognition, including weaker and competing reductions, is the goal: if a human finds a particular passage ambiguous, we would like for the system to share this perception.

Line Representation Summary

After the *General Properties of Frame Classes* section, the representation description begins with the features of notes and pairs of notes in a song context. These properties are applied first to the surface of a melodic line (the actual sequence that is played). Next, the features and emphases of the surface are shown to produce higher level lines (themselves sequences of notes) which are abstractions or skeletons of the musical surface. Each line is labeled by contours of features, grouped in regions, over its length. The properties of a simple sequence are recursively applied to the abstract higher levels. These sets of lines which constitute the representation of the surface sequence are described as being in a space of possible transformations which governs our perception of their similarity from different perspectives. The phrase is described as the containing unit of the lines of different levels; it has further features such as the start/end notes for each of these lines, and overall color determined by the relationships between these lines. Finally, the phrase is placed in a context of other phrases, and represented in terms of relationships to the proceeding and succeeding fragments, governed according to the conventions of the genre for structuring a solo.

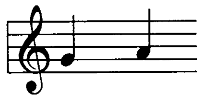


As a simple reference for the following sections, the diagram above shows the main frame classes in the representation of a melodic line. The arrows represent the "is contained by" or "is used by" relation: for example, NoteFrames are stored in PhraseFrames, and are used by pitch and rhythm category sub-agents to determine the color of regions in a sequence. These terms are explained in the following sections.

General Properties of Frame Classes

This representation of music is intended to reflect human understanding of music, defined in terms of the functional manifestations of this understanding. One of the fundamental measures of understanding is the ability to separate important from unimportant data. Each class below has a slot for *Emphasis*, the strength/salience in memory of the instance to the phrase it is a part of, and a slot for *Activation*, the temporary strength of the instance which determines its eligibility to be considered for matching (when listening) and its influence over the output (when generating). Each feature in each class below has an optional mechanism for determining its contribution to the instance's Emphasis and Activation. In addition, each class has a slot for *Color*, which describes the qualitative character of its instances; this slot is defined differently for each class[24]. In the later *PhraseFrames* section we will see how this combination of Emphasis and Color labels can be used to categorize melodic fragments. Forming abstracted lines requires these perspectives on what in the musical surface is perceived as standing out or being "marked for consciousness", in terms of the character and perceived accent of each event.

NoteFrames

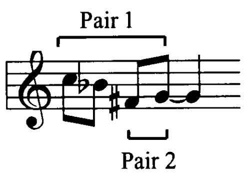


Example Slots	Note 1	Note 2
MIDI Pitch:	`67`	`69`
Duration:	`quarter`	`quarter`
Pitch in C:	`7`	`9`
Pitch in G:	`0`	`2`
Relation to BeatCycle:	`on`	`on`
Relation to HalfCycle:	`on`	`off`
Metric Tension:	`0`	`2`
Harmonic Tension (key of C):	`1`	`6`
Color:	`Landing`	`Passing`

The lowest level of melodic event is a single NoteFrame. A NoteFrame has an absolute pitch, an absolute time onset, a duration, and a loudness. A NoteFrame is also annotated by its relative pitch degree in the key of each of the following contexts: the song itself (the key signature); the local temporary harmony of the current chord in the song's chord changes; and the harmony of the succeeding chord. In addition, the NoteFrame's time is quantized with a position relative to the meter of the song, which places it in a specific measure and beat within a chorus. The quantized value allows the NoteFrame to be labeled as on- or off-beat with respect to increasing levels of a metric hierarchy[25]; the Metric Tension value is incremented for each offbeat level within a WholeCycle. These relative labels give the isolated note a degree of tension in the song context: off-beat times and unstable tones increase the overall tension of the NoteFrame. Each NoteFrame has a perceived Emphasis, which is the degree to which it stands out as important in the musical surface; this Emphasis is changed in the later sections by the relationships of the note to its surrounding notes. Finally, the NoteFrame has a Color, which is the computed label of its degree in the current chord relative to its emphasis: it is a Passing note, a Landing note, an Apog note (emphasized rhythmically but not harmonically), or an AltLand note (consonant harmonically but not rhythmically).

A single NoteFrame also has a set of pointers which indicate its roles in each of the structures described in the following sections; each structure can be queried from the NoteFrame to determine the specific role. In other words, a note contains accessors to information about the other notes it relates to in higher-level structures; these accessors can be used determine how it is heard according to these structures.

NotePairFrames



Example Slots	Pair 1	Pair 2
Pitch Interval:	`-5`	`+1`
PitchSeries step (key of C):	`TONIC-FIFTH`	`CHROM-SC`
Harmonic tension change:	`+1`	`-7`
Time Interval:	`dotted quarter`	`eighth`
Metric step:	`CrossCycle`	`EighthCycle`
Metric tension change:	`+3`	`+1`
Alignment:	`NormalAligned`	`ApogAligned`

A NotePairFrame in the system is a sequence of two NoteFrames. As described in the *Processes of Listening* section, a NotePairFrame is usually formed between two emphasized notes or two notes which are adjacent on the surface. It is labeled in terms of the absolute pitch, quantized onset time, and loudness differences between the NoteFrames. Each absolute difference (except for onset time, which is always positive) is further divided into two numbers representing separately the direction and the size of the interval. A NotePairFrame is also annotated by the placement of the pitch and time intervals in the harmonic and metric context: the pair is labeled as a scale step of each of the types of PitchSeries that it matches (a single pair can be in more than one PitchSeries), and as a CycleStep on any of the metric or ametric cycles that it matches.

The NotePairFrame provides information to separate NoteFrames in terms of the relative perceived emphasis of their features. The Emphasis of a more resolved tone is incremented, as is the Emphasis of relatively longer tones and louder tones. The relative Emphasis values of the NoteFrames for each feature may be compared to determine the Alignment color of the NotePairFrame. NotePairFrames in which a NoteFrame has a coincidence of emphasis from meter or perceived accent (velocity or duration) and a deemphasis harmonically are labeled as ApogAligned. NotePairFrames in which a NoteFrame has a coincidence of an emphasis from perceived accent and a deemphasis metrically are labeled as SyncAligned. All other alignments are considered NormalAligned. A NotePairFrame Color label contains both this Alignment label, and integers for the change in Harmonic Tension and Metric Tension between the notes; these correspond to the intuitive concepts of a decrease in tension with tonal resolution and in movement from an upbeat to a downbeat.

NoteSequenceFrames

A NoteSequenceFrame is a fundamental data type for a melodic line, used on multiple levels for each actual surface line. A NoteSequenceFrame is simply a list of nodes which each contain a NoteFrame, such that each NoteFrame is in a NotePairFrame with the succeeding note. NoteSequenceFrames apply to notes in succession (notes which are proximate, either touching or with other notes in between). The NoteSequenceFrame is labeled by ContourFrames (as described in the next section) for each of the features of the NotePairFrames over its range. Each node has a list of pointers to previous and succeeding NoteSequenceFrames which use this node, serving as possible branching points. A traversal of a NoteSequenceFrame (for comparison or generation, as described in the *Processing Music* section) can then follow any of these branches.

A NoteSequenceFrame has slots for its level and the number and position of its constituents. A NoteSequenceFrame has an additional calculable annotation for the quality of the NoteFrames in its first and last position: a first note that is of high tension harmonically or metrically carries a label of Abruptness, and a last note that is of high tension metrically or harmonically carries a label of Unresolvedness. For these cases, the non-surface level NoteSequenceFrames use the nodes' branching lists as pointers to alternative start and end notes; for example, these can be notes which do not fit directly in the cycle position expected by the NoteSequenceFrame, but which provide the tonal resolution that is expected (although in some cases there are no such alternates). This allows for the traversal of the NoteSequenceFrame and its comparison to other NoteSequenceFrames that are similar except for these starting and ending distortions.

The NoteSequenceFrame has higher Emphasis of its first and last NoteFrames, roughly corresponding to the perceptual effects of primacy, and recency, and change (the change to begin a Region or to discontinue it) in gestalt cognitive processing. The NoteSequenceFrame agent's actuators have the ability to project an instance's continuation either forwards or backwards. That is, a NoteSequenceFrame can calculate what would be its next event, or what would have been its preceding event, by assuming continuation of each of the features in the first/last NotePairFrame. This information is used to generate expectations when listening, and to generate possible next notes when playing.

RegionFrames

A RegionFrame defines a label of a NoteSequenceFrame over the section of the ContourFrame in which it applies. RegionFrames each contain one or more instances of a specific feature value, defined at each NotePairFrame over the range of the NoteSequenceFrame. A series of similar feature values are placed in a single RegionFrame. Through the RegionFrame construct, the features of NoteFrames and NotePairFrames suggest a grouping of like features. For example, repeated pitches of 'F' are placed in a RegionFrame (of feature: NoteFrame->pitch = F), and consecutive pairs of notes ascending a minor scale are placed in a RegionFrame (of feature: NotePairFrame->scalestep = MINOR-SC). The Emphasis of a RegionFrame is simply its length (longer regions are stronger); the Color is the feature that it represents. The feature represented by a RegionFrame can be an abstract category, such as a type of rhythmic figure category, as well as a specific label.

RegionFrames which have constant time intervals reflect portions of their NoteSequenceFrame as a cycle; RegionFrames which have constant PitchSeries step-motion reflect portions of their NoteSequenceFrame as scale movement. A single NoteSequenceFrame can then be seen from several perspectives, for different features at overlapping positions along its range. Through this mechanism, a NoteSequenceFrame is able to reflect properties of a sequence of pitches (such as a PitchSeries) as well as a sequence of timed notes (such as a MetricCycle); thus, every salient sequence of pitches is automatically evaluated as a sequence of timed notes, and every salient sequence of timed notes is automatically annotated as a sequence of pitches.

ContourFrames

A ContourFrame is a set of RegionFrames. A ContourFrame has a simple rhythm, defined as the temporal intervals between the starting point of each new region, and a tempo, defined as the distance covered (amount of change in feature value) over time. That is, the relation between successive elements in the NoteSequenceFrame is captured literally by the RegionFrame, but the changes in the features values which break these regions create the rhythm and tempo of the ContourFrame: the ContourFrame emphasizes the discontinuities/changes in feature values. For example, the ContourFrame's rhythm can express alternation between tense and stable elements (as in meter with onbeats and offbeats, or an abstract sense of direction and return in a phrase). These values of rhythm and tempo are realized through the Emphasis of the events which occur on the RegionFrame breaks, causing these events to be on a higher level NoteSequenceFrame which has its own Color categories for describing rhythm (see next section). Through these values, the ContourFrame serves as the label of the NoteSequenceFrame's abstract rhythm and tempo with respect to each feature; for example, the ContourFrame indicates if the NoteSequenceFrame is a smooth or jerky pitch line, the overall direction of the pitch line and its trajectory towards its goal (expansive or constricting motion), and the speed with which it covers registral or harmonic ground.

Given a NoteSequenceFrame, correlations between the ContourFrames for each of the features can provide information about the Color and Emphasis of the NoteSequenceFrame. Those sequences of events which contain extended RegionFrames for more than one feature are considered salient (strong in memory); thus, their Emphasis is increased. In addition, NoteSequenceFrames with extended regions for Alignment values that are NormalAligned are emphasized.

Comparison of NoteSequenceFrames requires comparison of their contours as well as their other slots. For each feature described over the sequence, the similarity of contours is defined as the average value of the regions for that feature, combined with the number of matches in exact regions and region breaks. That is, one element of the NoteSequenceFrame similarity is defined as the difference of the overriding feature colors of the NoteSequenceFrames; another is the number of specific matches between feature regions and their spacing in the contour. Two regions within a NoteSequenceFrame's ContourFrame can be compared in terms of their feature value. The type of value of similarity depends on the feature; the difference between scale types (arpeggiation and minor scale steps) is an integer, whereas the difference between an ascending and descending line is binary.

NoteSequenceFrame Color

The NoteSequenceFrame agent has a list of potential categories of feature motions in its ContourFrames which determine its Color. Each of these categories is like a sub-agent, consisting of a miniature template and a label; if an instance fits the template, it is put in a RegionFrame and assigned the label. (By default, sub-agents simply check the value of the RegionFrames and use these labels, which are the values for the constituent NoteFrames and NotePairFrames.) A NoteSequenceFrame can also detect an unusual feature in its constituents: either a leap or pitch movement that is not in a scale, or a misalignment of Emphases. Examples of categories to handle the conventional occurences are below; when no category is available, a generic Weird label is assigned by default (to indicate that the feature is not expected in this genre).

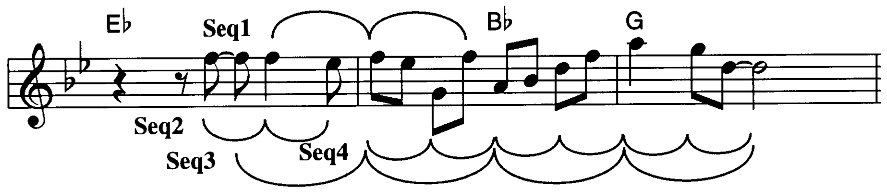
Pitch motion that fits within a PitchSeries is considered standard, and is labeled as an instance of the PitchSeries. Unusual pitch motion is labeled by whichever of the following categories matches: the arpeggiation of a substitute triad (labeled SubTriad); an unprepared chromatic leading tone to a resolved tone (labeled UnPChrom); the suggestion of two voices instead of one (labeled TwoVoice - currently unused). More common non-PitchSeries motion is a simple Leap, which is further labeled GapFill (after Narmour) if followed by a smaller interval in the opposite direction, and labeled Squeeze if successive pitches are on alternate sides of a goal tone and are increasingly close to it. [Dowling 94] presents evidence that suggests that pitch contours are a salient feature of simple melodies; here, this is accommodated directly with the use of ContourFrames and specific labels for types of contours.

The rhythm contour corresponds to the presence or absence of each of the higher metric cycle levels at each smallest CycleStep along a NoteSequenceFrame[26]. Regions in which the strong beats are naturally emphasized are labeled *OnBeat*. While this is the basic definition, the deviations from the expected patterns are somewhat more complex. Several categories of deviations are generally referred to as creating a feeling of syncopation. The standard definition of syncopation is of relatively greater stress on a weak metrical beat, or a lack of stress on a strong metrical beat; in the system, these are labeled *OffSync* and *AbsSync*, respectively. However, the stress referred to by this definition can be of several types having to do with the alignment of features (as discussed in the *NotePairFrames* section), and can also have a varying degree. By default, it refers to a velocity accent, but can also be realized through any of the other types of Emphasis discussed here, such as harmonic resolution or a phrase closure (these are described in the *PhraseFrame Color* section). Repeating ametric cycles are labeled as CrossCycles, with phase and period features in RegionFrames. In addition, an entire region in which a syncopation applies suggests an alternate cycle, and is labeled *AltCycle* in a RegionFrame; this refers to an implied shift in phase of the default meter, as opposed to a change in period of a suggested ametric cycle. These rhythm patterns and deviations can occur on any of the default metric levels; thus, each metric level NoteSequenceFrame (described in the *Levels of NoteSequenceFrames* section) indicates whether it is syncopated in a given region.

This definition defines several types of rhythm colors as standard categories which are common in this genre. These rhythm colors are simplifying concepts with which to consider similar rhythmic figures as being of the same sort. The most common of the categories described above are: playing on the beat with natural emphasis (OnBeat); playing with abrupt start/end notes (AbsSync); playing with syncopated emphasis (OffSync); playing a dotted quarter cross rhythm pattern (CrossCycle); and emphasizing the offbeat elements of a cross rhythm pattern (CrossCycle and OffSync). Further categories not covered in the current design include the use of timing deviations from the quantized slots, such as the type of rhythmic figure which flattens eighths into figures similar to triplets.

Levels of NoteSequenceFrames

A single melodic line is represented as several levels of NoteSequenceFrames on top of each other. The first level is simply the musical surface, the list of notes as they are played in; the later levels are successively higher skeletal abstractions of implied structures in this surface. There are default abstract levels for each level of the metric hierarchy (i.e. BeatCycle, HalfCycle, etc.). There are also abstract levels for additional NoteSequenceFrames with ametric cycles in their collections of emphasized notes. For every NoteSequenceFrame in which there are successive and non-adjacent events with high emphasis, a new higher level NoteSequenceFrame is created between the more emphasized events. The diagram below indicates a few strongly emphasized NoteSequenceFrames in a melodic line that have nearly constant cycles for their range: **seq1** is a CrossCycle; **seq2** is a shifted BeatCycle; **seq3** is a HalfCycle, with alternate branches of the start/end notes around the exact half note positions; **seq4** is a BeatCycle. NoteSequenceFrames can be any sequence of notes, regardless of whether or not they form a continuous metric cycle or PitchSeries traversal; however, those NoteSequenceFrames which do have these properties result in higher Emphasis (from the RegionFrame lengths).



The abstraction of a level of NoteSequenceFrame from a lower level is a recursive process. The number of possible NoteSequenceFrames which can arise through this process is large. Thus, in the system each NoteSequenceFrame is given an Emphasis corresponding to how perceptually salient it is in terms of its component features and their correlation. Only strong NoteSequenceFrames are effectively considered to be the representation of the melodic line; the processes of listening and generation are only influenced by strong Frames[27]. A series of unusual features decreases the relative emphasis of a line ordinarily; but when a line is otherwise strong, an unusual feature interpreted as an important intentional character of the phrase.

NoteSequenceFrames as Reductions

Analytic reductions of music are high level musical structures; they are interpretations through which some notes can be heard as serving subsidiary roles to others. In particular, some events are said to be elaborations or ornaments of major events (called *heads*), or are said to lead up to or prolong major events, recursively through larger sections of music [Lerdahl and Jackendoff 83][28]. In another sense, a reduction is a simpler statement of a phrase that "sounds like" the first in terms of capturing its structurally important progression of notes[29]. In my system, reductions are apparent as strongly emphasized NoteSequenceFrames between events not necessarily adjacent on the surface. For example, the abstract NoteSequenceFrame which connects the most emphasized events in successive FeatureMatchRegions is likely to correlate to reductions commonly found in music theory. Similarly, those NoteSequenceFrames which have several correlated feature RegionFrames effectively combine the evidence of these features in adding to the strength of the NoteSequenceFrame, making it a likely reduction.

The comparison to music theory terms would further suggest that for a given NoteSequenceFrame, the less emphasized events are prolongations of or progressions from the more emphasized events (which would appear on a higher level NoteSequenceFrame). Each of the events on the high level is seen as the head of a figure which includes events on the lower levels; the lower levels can be segmented into (possibly overlapping) sets of elaborations of higher levels[30]. The realization of a line from the high level requires recursion down a structural tree to find the properties of the lower levels. From this perspective, each line can be seen as a set of destinations and realizations of paths towards these destinations. The destinations are the structurally important anchor points of the phrase, and the paths are the particular coloration which this skeleton is ornamented with. Thus, the possible lower levels constitute various elaboration mechanisms of the higher levels. The transformation of a basic tonic note played on a downbeat, for instance, can include its elaboration by eighth notes which turn the single note figure into an appoggiatura ornament. The set of emphasized events in a phrase serves as a set of anchors which determine the phrase's general shape without regard to the various possibilities for the less emphasized material.

PhraseFrames

An actual melodic line, then, is represented as the collection of the surface line NoteSequenceFrame and each of the NoteSequenceFrames abstracted from the surface line, each of which has their own features and Color labels. The container of this information is the PhraseFrame, which also holds lists of the constituent NoteFrames and NotePairFrames; each of these lists of instances is sorted according to Emphasis. In this genre, the phrase breaks are usually clearly demarcated and defined by the conventions of solo structure (described in the *SoloFrames* section), which makes this a convenient category to use here: rather than simply referring to the collection of NoteSequenceFrames, we can refer to this collection as a phrase with its particular features.

The PhraseFrame has slots for each of the levels of NoteSequenceFrames. Its labels consist mainly of accessors to the labels of the constituent levels; for example, a phrase's alignment with the meter is determined by each level's correlations of alignment features. Each level, then, is roughly analogous to a Level-Band [Minsky 86] of specificity relative to the surface of the phrase; where the surface of the phrase provides the specific values, the higher level lines are the general type of movement. Within a PhraseFrame, those aspects that are at the appropriate level of specificity for the current analysis/generation task are used. The PhraseFrame is also labeled in terms of its location: position in the chorus/solo, and the position of the head (most Emphasized element) of the highest level relative to the meter. Finally, the PhraseFrame is labeled in terms of its reference to other phrases; this is indicated by a list of pointers to PhraseRelationFrames for those other instances which a phrase's constituents strongly match.

PhraseFrame Structure

The RegionFrames of NoteSequenceFrames provide the basis through which parallelism within a phrase is defined. The structure of a NoteSequenceFrame is a set of FeatureMatchRegion pointers, each of which is a RegionFrame of successive FeatureMatchEvents. A FeatureMatchEvent consists of a pointer to an original event (a NoteFrame or NotePairFrame) and a repetition event, and is labeled according to the difference in the events' features. Any feature can be compared, ranging from pitch value to emphasis alignment. The Emphasis of a FeatureMatchEvent is incremented for every feature whose value is the same in these two events; FeatureMatchEvents with high Emphasis also increment the Emphasis of their constituent events. Thus, the FeatureMatchRegion is analogous to the RegionFrames: it indicates a region over which the property of matching a specific previous event/region holds.

The structure of a phrase is then the set of the structures of its constituent NoteSequenceFrames. Intuitively, the structure of a line is the perceived repetition and variation of short fragments (see diagram in

Intuitive Description section). Here, this is literally implemented; the FeatureMatchRegions of high Emphasis indicate positions where a PhraseFrame repeats itself in some features, and perhaps allows the others to vary. By only examining the FeatureMatchRegions for their positions and not their specific feature values, the structure of the PhraseFrame as an abstract feature is obtained and can be compared against the structure of another PhraseFrame . In other words, the system can compare the underlying form of two lines, without any reference to specific features; this allows for a representation of how solos are structured abstractly, as is discussed in the *PhraseRelationFrames* section.

PhraseFrame Color

The character of a phrase is a complex and multifaceted quantity. Therefore, it is represented in terms of several different values. In the terminology of SOM, the constituent saved instances are K-lines which represent the saved states of agencies concerned with different aspects of the Color of a PhraseFrame. The Color of a PhraseFrame can initially be considered as the sum of the descriptors of each of the NoteSequenceFrames, scaled according to their Emphasis. For example, the PhraseFrame's overall syncopation is determined by the scaled sum of the syncopations of the NoteSequenceFrames. Further, the presence of each level of NoteSequenceFrame is itself a descriptor of the PhraseFrame's Color: whether or not a phrase has a CrossCycle, or an emphasized HalfCycle pulse, for example.

The distinct nature of the phrase, or the aspects of it which are important, are captured by the sorted lists of its constituent NoteFrames, NotePairFrames, and NoteSequenceFrames; the more emphasized instances are earlier in the lists, and reflect the important properties of the phrase. These instances then are the ones which are used in determining the similarity to other phrases, or in guiding generation of variations of the phrase. Broken expectations or unusual features in prominent positions are part of these important instances, because they are emphasized by the strong features which make them prominent. For example, peaks in the pitch contour are discontinuities of direction which are given an Emphasis as a change of RegionFrame feature value, and which thus become distinct features of the phrase. These elements of a phrase's character which result directly from the processing of it are described in the *Processes of Listening* section.

The particular characteristics of the end of a phrase are important because they contribute greatly to the overall perception of the phrase. Just as each of the individual NoteSequenceFrames can end either abruptly or with resolution, the PhraseFrame itself has a quality of ending consisting of the sum of its NoteSequenceFrames' endings. When several of the levels end with resolution on the same note, this produces a quality of closure; the degree of this closure varies depending upon the endings of the NoteSequenceFrames'. Also, the overall direction of a phrase in several features affects its closure; for example, phrases which decrease in pitch register, harmonic tension, and rhythmic activity have an increased feeling of closure. This quality then constitutes a formalized description of the intuitive sense that some phrases are "opening" gestures and others are "closing" gestures; some phrases end in a climax while others fade out, some land on a final beat while others remain unresolved. As the *PhraseRelationFrames* section describes, these overall qualities of phrases are then used in determining the structures between successive phrases.

In addition to descriptions of surface contours and individual level's features, we would like to be able to categorize general perceived qualitative effects of a phrase; these can be defined as combinations of the individual features. In the current design, one example has been developed. We can describe the general perception of *motion* occurring over regions of the phrase; the motion relative to time then gives a sense of the perceived tempo (referring to amount of activity, not metronome tempo) of the phrase. With this representation, the concept of motion refers to the same quality of several distinct levels. A series of notes in a sequence is considered to have greater motion for lower metric levels (e.g., eighth notes) than for higher ones (e.g., whole notes). The same property holds for series of chord changes relative to metric level; but chord progressions also have a feeling of greater motion if they covers a greater harmonic distance (as defined in the *Harmony* section). Further, the density of the patterns found in the melodic line representation, and the redundancy of the closure in the phrases, yield a quality of motion. The change from distorted (e.g., harmonically tense or misaligned) regions to undistorted regions creates a perceived motion which can be altered without changing the underlying surface rhythm but with a definite change in perceived activity. Each of these phenomenon contributes to the overall value of the PhraseFrame's motion.

Overall labels of the character of a phrase are difficult to define, but are important for understanding music. The detailed possibilities for these labels have not been filled in extensively in the current implementation, which instead uses PhraseFrame Color as a slot-holder in order to show where in the representation such concepts may reside for further work. The important point here is that the representation supports the above types of discussion about the character of a phrase; with the basic terms defined, questions of character and affect can be further explored.

Comparing PhraseFrames

In order to clarify how the PhraseFrame corresponds to an intuitive concept of a simple musical line, this section explains the features used for comparing two phrases (thus describing the theoretical space of possible PhraseFrames). Similarity between two phrases is not a simple binary value; it is an exhaustive listing of the similarities and differences between features. There are many possible predicates which may be defined on a data structure and used to determine features for comparison, but only a subset of these is salient and should be relevant in our representation. Humans' conception of music has considers some melodic lines to be more similar than others; in the system, the coefficients (control variables) for determining how much each type of feature difference matters in determining the overall similarity can be changed. The opposite side of comparison is generation of variations; as we will see in the *Processes of Playing* section, these similarity metrics can be viewed as possible transformations of a phrase to produce variations.

Comparing phrases can occur with respect to the following features: member-wise comparison of NoteSequenceFrame levels; PhraseFrame Color values; the relation of the constituent NoteSequenceFrames to harmonic and metric context; overall distance and amount of motion covered in time, pitch register, and harmony; start/end times and pitches; and presence of similar fragments. This allows for PhraseFrame matches of both absolute and relative properties; phrases of similar length and phrases which start on the same notes are examples of the former, while phrases with similar metric position (for each metric level) and phrases which ascend from whatever note they start on are examples of the latter. This also allows for PhraseFrame matches of both specific and abstract features, such as the difference between a match of a missing metric slot compared to a match of a general syncopation. Examples of phrase comparisons are given in the *Example Comparisons* section.

PhraseRelationFrames

A PhraseRelationFrame holds information about successive phrases, including the information that arises from the above comparisons, as well as higher level concepts. The relationship between successive phrases yields information that can be used to understand the high level structure of an analyzed solo, or to generate one. This section is rudimentary in the current implementation and design, existing only to stake out the position in the architecture for such work. The framework presented here provides the terms with which to define these relations, such that new concepts can be developed and tested, and the basic musical notions with which to regulate the actual construction of these structures. For example, the system notices the pitch interval between successive segments, and can thus detect the surface discontinuity between two phrases which the high level structures do not address.

Currently, a few types of relations are calculated for PhraseRelationFrames. Each PhraseRelationFrame is labeled in terms of the change in Color between phrases, which is a separate value for each type of color described above. The surface distance between the end of the first phrase and the start of the second phrase indicates the smoothness of the transition; the distance between the emphasized heads of the phrases indicates the amount of motion in the transition. The only symbolic labels which are currently defined for PhraseRelationFrames are CallAndResponse and StatementAndVariation structures: successive phrases in which the first is an opening phrase and the second is a closing phrase (according to the PhraseFrame Color described above) are labeled as CallAndResponse ; successive phrases in which the second matches a significant number but not all of the features of the first phrase are labeled as StatementAndVariation.

SoloFrames

A SoloFrame is the top level structure for a melodic line, describing the script-like progression of an entire solo. The SoloFrame uses a chorus form as its default metric structure for a single solo. Just as the song's metric structure is labeled by the chord changes over the course of a chorus, a SoloFrame labels its metric structure by the properties of the melodic line over the course of a chorus. Intuitively, a SoloFrame should be seen as a holder for scripts or descriptive commentaries about solos: the structural relations and corresponding characteristic quality of a SoloFrame allow for a description of rising intensity, or trajectories through "moods", or an entire a dramatic form played over the course of a chorus.

The most important property of the melodic line over the chorus is the structure relating successive phrases; thus, the SoloFrame indicates the form of the solo with a list of PhraseFrames. The PhraseFrames need only be partially specified: a structure with a position, a color (absolute values for any of the PhraseFrame features), and relations to the prior phrases (through PhraseRelationFrames). The SoloFrame indexes this list by numbering the phrases from the start of the solo, and using this to organize the PhraseRelationFrames.

The default conventions for the structure of a solo within this genre are represented in static SoloFrame instances called SoloConventions, each of which is an incomplete specification of slot values which reflect a particular solo characteristic. There are a few separate types of solo structure apparent in the studied solos of this genre; these are reflected in a set of default SoloConventions which share some features and conflict in others. The current SoloConventions are as follows: the placement of phrases at the start of every fourth measure, or at the start of every second measure; the division of a thirty-two bar form into two sections, each of which uses the AA'BA form (as is common in song's tunes, where AA' is a StatementAndVariation), or the twelve-bar model (roughly ABA'BCB'); the presence of CallAndResponse relations between pairs of successive phrases; relative increase in motion and tension values through the third phrase in a sixteen bar segment, and then a decrease in these values leading to a strong closure in the final phrase. Any of these SoloFrame instances can stand alone and minimally constrain its contents, or several can be active simultaneously to create a generic solo structure.

The final static SoloFrame instance in the system is a special instance reserved for holding input from the user (called UserSolo). This currently is simply a place holder for settings of features that allows for the user to directly influence a solo; however, it is through this structure that input controllers could be attached to the system if it were to be used as an instrument.

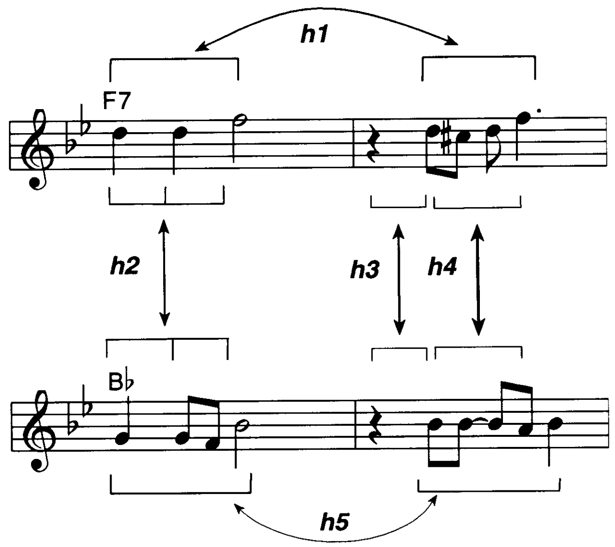
3.3.4. Example Comparisons

The above lengthy description of the details of the representation of a melodic line is confusing if not considered in the light of concrete tasks which reveal its utility. One benefit of this representation of melodic line is that it can be used to compare phrases. The *Processing Music* section describes how the system actually performs these tasks; here, I will show how the representation supports these processing mechanisms. These examples are intended to be contrasted to the limitations of traditional music representation schemes, which cannot express these types of phrase comparisons. I believe that these examples clearly indicate the utility of the representation, and thus verify the efficacy of my approach.

The point of these examples is to show which similarities and differences between phrases various methods of comparison and representation provide, so that the reader may determine the accuracy of these approaches at concurring with his own intuition. For each pair of phrases being compared, their similarity is calculated separately for each feature. In order to determine that a pair of phrases are similar, the system has to first recognize those features which constitute their similarity. Thus, comparing phrases is one evaluative test of a representation that reveals its ability to capture the salient features that humans perceive.

Each feature is a generalized concept, such that it can be recognized in many different instances which have other conflicting features present; we say that the system is able to *generalize* the concept *over* the differences in the other features. The other side of this is that the system is able to notice which are the changed features in contrast with those that stay the same. These ideas combine to give the system its sense of what is important in a pair of phrases: the repeated features are seen as more structurally important, while the different features are seen as more important for local characteristic color[31].

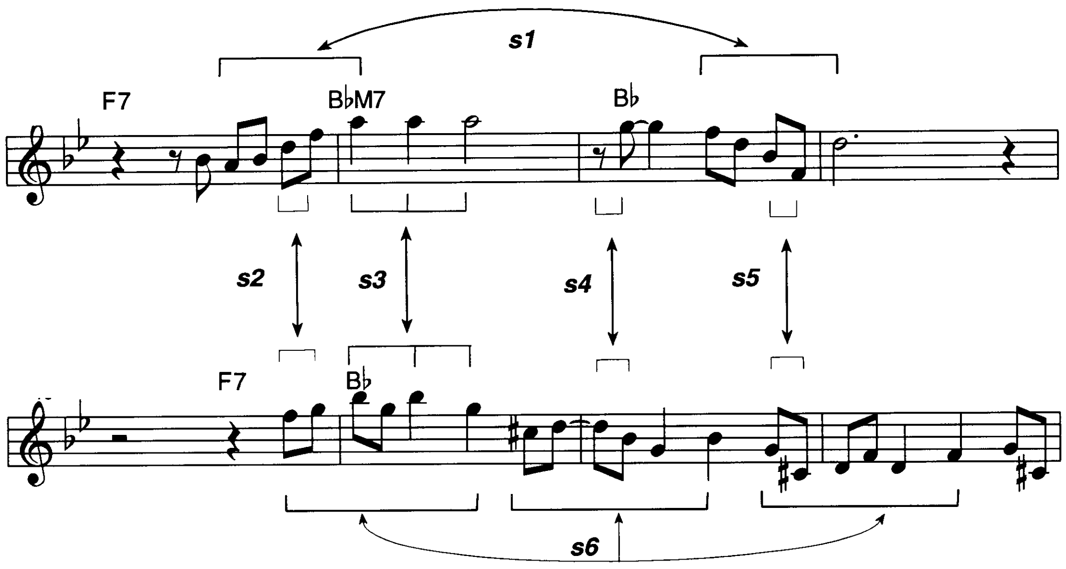
Example Comparison #1



Here is a transcription [Schiff 61] of Armstrong's performance of the first four bars of the song "Heebie Jeebies". The fragments delineated by brackets and connected by arrows indicate emphasized perceived similarities. This selection consists of two two-bar phrases, each of which is itself two one-bar phrases. The system recognizes matched line fragments here within each of these phrases and between them. BeatCycle and HalfCycle NoteSequenceFrame abstractions of measure one (M1) and M2 have a similar contour (***h1***); the BeatCycle has a repeated D and then a MAJ6-ARP step to an F, while the HalfCycle just has the step to an F. The extra offbeat eighth note (C#) in M2 does not affect the higher level cycles, which can still match M1 even though the surface level is different. Note that even though the relative metric position of these NoteSequenceFrames has been shifted, the match is still possible; further, the final F in M2 is matched to each of these cycles (through an alternate branch node) even though it has been shifted an eighth early relative to the new position. In other words, because the representation can generalize cycles over metric positions and syncopations, the system is able to notice this similarity.

Between M1 and M3 a match also occurs (***h2***), here by virtue of the ability to generalize pitch contours relative to the current chord over transposition, and also again abstracting away from the surface difference of an additional eighth note. In addition, we can use the structure of M1 and M2 to help us understand M3 in relation to M4. Since we have already identified that the PhraseRelationFrame beginning in M1 matches the phrase beginning in M3, we can expect a continuation of some features of this match. The structure of the phrase in M1 and M2 consists of a statement in M1, and a variation of this statement in M2 (including a shift of metric position and an alteration of the surface rhythm). We can thus match the first phrase to the second phrase based upon their abstract repetitional form in the PhraseRelationFrame, which suggests that M4 will be a variation of M3 (***h5***). Indeed, the same metric shift of position and emphasis occurs in M4 as occurred in M2 (***h3***), and there is also a surface rhythmic alteration. The presence of a cross rhythm (***h4***) emphasis is here produced by two succeeding groups of eighth notes, whereas in the M2 it was produced by the beginning and end of a four eighth-note phrase; however, the representation generalizes the concept of cross rhythm to encompass both of these, and to notice that they are the same period and phase relative to the measure. M4 includes other differences form M3 as well, including a transformation of the pitch contour, and the absence of the second quarter note.

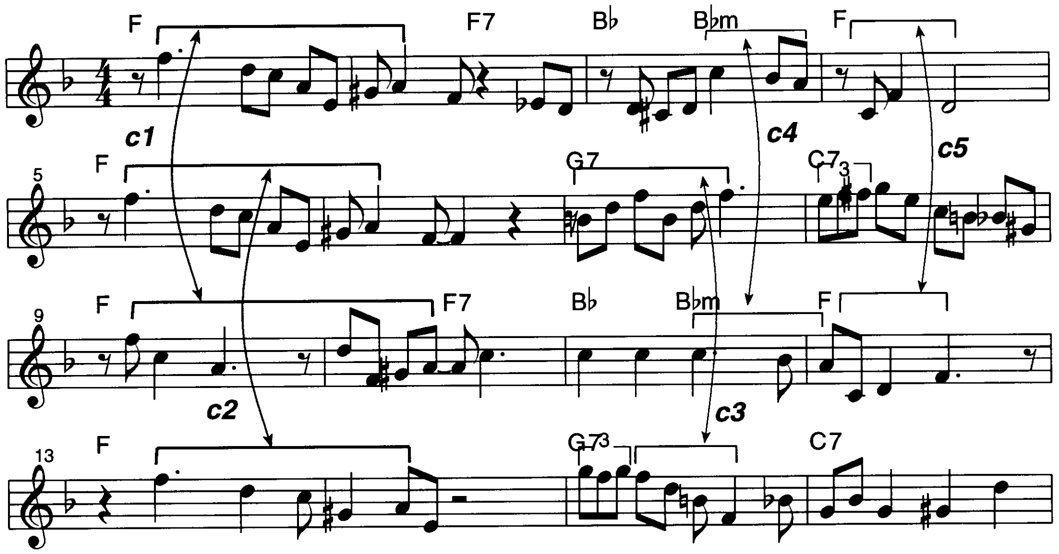
Example Comparison #2



Here is a transcription [Schiff 61] of two phrases from Armstrong's performance of "Struttin' with some Barbecue"; the first is the opening line of the tune, the second is the opening line of Armstrong's improvised solo. ***s1*** and ***s6*** are similar chunks repeating within a single phrase, while the others are similar chunks between the two phrases. As in *Example Comparison #1*, these similarities are the result of similarities between features which are present in the representation of the phrases.

s1 matches two NoteSequenceFrames of four eighth notes: they both are arpeggiation RegionFrames located in the same metrical position, with the direction of pitch intervals inverted. ***s2*** and ***s5*** generalize metric position and interval direction over interval size. ***s3*** requires using a higher level than the surface (a BeatCycle), and generalizing from rhythm similarities over pitch differences (repetition is transformed into chord arpeggiation). ***s6*** matches the phrase constituents of a two eighth-note pickup and an abstracted three quarter-note figure (as in ***s3***), where the second instance has deleted the first quarter note, and the remaining section of the figure is inverted; the system notices that this figure has a similar metric position with respect to the start of each measure. ***s4*** requires that both types of syncopation (a suspended prior note or a prior rest) are labeled as being of the same general category of syncopated phenomena.

Example Comparison #3



Here is a transcription [Schiff 61] of four phrases from Armstrong's performance of "Cornet Chop Suey"; the first two four-bar phrases (M1-8) are played in a chorus early in the song, while the second two four-bar phrases (M9-15) are played in the same position in a later chorus. The system uses both metric position and current/succeeding chord values to determine that the compared melodic fragments occur in similar contexts. Whereas the previous examples indicated more precise matches of rhythms, etc., this example demonstrates how less exact matches are possible.

c1 and ***c2*** match high level (i.e. roughly WholeCycle, with branches) NoteSequenceFrames' start and end pitches, the syncopated metric positions of these notes, and the general contour of descent followed by ascent through a G#. These phrases are thus substitutable for each other in terms of their general role in the tune, and the system is able to notice this abstract similarity despite the many surface differences. Similarly, ***c4*** matches the pitch line (C Bb A) between two different rhythmic surfaces. The ContourFrame comparisons allow matches of regions that have similar features, such as pitch color. ***c3*** matches both the arpeggiation of a G7 chord, as well as a CrossCycle of emphasized pitches. ***c5*** indicates a match of averaged pitch color region (MAJ6-ARP) in a ContourFrame.

3.4. Processing Music: Representing Musical Behavior

In addition to knowing what sequences and phrases are, we must know how they are created and used by processes. The way the system demonstrates its ability to understand music is through listening and generation tasks; the *Example Comparisons* section is an example of a listening task that is required for understanding. Part of understanding music is knowing how to produce and parse it; therefore, the processes in this section constitute a portion of the system's understanding of music.

Given a network of concepts and instances, these processes can be loosely described as the manipulation of these structures through their consecutive consideration and alteration. The mechanism of spreading activation through a network [Maes 94] is intuitively appealing as a model for these processes of thought and behavior. The processes described here are roughly inspired by spreading activation networks, in particular to combine evidence from different sources when listening and generating.

3.4.1. Processes of Listening

The goal of listening is to form an understanding of an input solo line. In the context of this system, forming an understanding means translating the incoming MIDI stream into instances (saved in memory) of the representation above; the list of timed notes is translated into symbols of types that are previously understood[32]. An important aspect of listening is noticing references and structure. For example, the listening processes described below are the means through which the system is able to notice the features indicated in the above comparison examples.

The power of the representation becomes clearer here, when seen in terms of the knowledge that it provides for analyzing input. The system is able to extract skeletal lines (reductions) from the input, it can categorize regions in terms of their color and melodic motion, it can form expectations and make predictions about the continuation of the input, and it can notice odd or distinctive features of the input. These perspectives constitute the system's memory of the input line; the recognition of them constitutes a successful analysis.

Beyond simply an algorithmic translation of notes into representations, the processes of listening described below are meant to be models of the behavior of listening, including aspects of that behavior which result in "confusion" or "disappointment". By basing the processes here upon a model of listening behavior, features such as the comprehensibility of input lines can be represented. As the processes below operate, they send status messages to the interface reflecting their progress in the analysis; currently, these are printed on the console, but they will be added to the graphic interface. These status messages reflect: the creation of expectations and ensuing disappointments; the degree of comprehensibility of the input in terms of its feature continuity, the Emphases of the created instances, and the matching of high level features (such as a CallAndResponse form, or a PhraseFrame with high closure). (More such status messages should be added.) The system also notices the relative importance of features; the results of these processes can be seen by examining the resulting PhraseFrame's lists of Emphasized instances. While these processes are of course an oversimplification of actual human cognitive processing, it is still advantageous to approach the construction of a system from this perspective, in order to ensure that the design decisions are motivated by the nature of the task rather than implementation efficiency.

Creating Instances

As the MIDI driver receives input notes, these are turned into instances of the representation. Each agent's *sensors* detect and label instances of it's corresponding concept in the input; this is the basic mechanism through which translation of a note into the representation occurs. (Recall that each of the frame classes above corresponds to a musical concept which is managed by an agent; the instances of the representation are the saved slots for the frame, filled in by the agent.) One perspective of the hierarchical arrangement of the agents is defined by the paths between each agent and the relatively lower-level agents which fill in its slots. In the system, as each agent notices its own occurrence, it triggers the agents which use it to fill in their slots to look for their occurrence[33]. Thus, the basic listening process is a recursive event-driven loop based on input notes; this is combined with processes based on time delays described in the *Expectations* section.

A new SoloFrame is created when the listening task begins; this constitutes the listener's readiness to analyze a solo. A new PhraseFrame is created when the first note enters, and new PhraseFrames are created when the analysis determines that the currently active phrase has ended (described below). All new instances (of the other frame classes) created from the input are placed in the currently active PhraseFrame. By default, the PhraseFrame agent creates new NoteSequenceFrames for the musical surface, and for each of the metric cycles.

As described in the *Representing Music* section, each instance of a frame has an Activation level, which indicates its temporary availability for processing, and an Emphasis level, which indicates its overall strength in memory or importance. As each instance is created below and added to the PhraseFrame's lists, these levels are set by the agents that create the instance. By default, a new instance has a constant boost of Activation (to correspond to a recency effect), and further increases of Activation in proportion to its Emphasis (each increase of Emphasis also increases the temporary Activation). The Emphasis is determined separately for each type of instance. While the Emphasis is a long-term value changed only by the agent that creates the instance, the Activation is consistently decremented by a constant amount at the input of each new note and at the occurrence of a slot in each metric cycle. This corresponds to the intuitive idea that the degree of resonance in short term memory of an event is decreased as other events are attended to or just as time passes; several of the processes below work by checking memory to find instances with temporarily high Activation. The intended consequence of this phenomenon is that the structures which are created to represent the parsed line reflect their relative importance in the music.

In this scheme, many NotePairFrames and NoteSequenceFrames are created[34]. While all of these possible links are kept initially, only those with strong emphasis are maintained. Since each of the lists of instances kept in the PhraseFrame are sorted according to Emphasis, the threshold (in the control variables) for determining the maximum length of these lists (or the minimum acceptable Emphasis) can be changed to determine how much detail is remembered about the input. (Most of the connections between notes are not important, and do not have a high Emphasis; these connections should be forgotten and regenerated later, if needed.) This corresponds to the intuitive idea that material which has perceived emphasis is more easily remembered; here, notes which are accented have higher emphasis and are available for matching with other notes, and thus contribute to the accessibility of their enclosing NotePairFrames and NoteSequenceFrames. This mechanism effectively constitutes competition between the different instances of an agent.

Labeling Notes and NotePairs

On the lowest level, each note is translated into a new NoteFrame instance by simple assignment of the MIDI values to the pitch, loudness, and time slots in the frame. Then the NoteFrame agent polls the demon to determine the current position in the song chorus, and uses this information to assign the slot values relative to pitch and meter. The NoteFrame's Activation is given a constant boost to allow it to be considered in relation to succeeding notes, and it is added to the current PhraseFrame.

Once a new NoteFrame's slots are filled, the NotePairFrame agent is triggered to check if the note should be paired with any other notes. Each of the NoteFrames in the PhraseFrame's list are checked to see if their Activation is above a threshold; if so, a new NotePairFrame is created between the old NoteFrame and the new NoteFrame. (The new NoteFrame can be placed in several NotePairFrames with different old

NoteFrames). Then each of the slots of the NotePairFrame are filled in by comparing the values of the two NoteFrames, and the Alignment values are determined. If the change in the velocity, duration, or tension values are above a threshold, the Emphasis of the louder, longer, or more resolved NoteFrame is incremented. This design allows the basic NotePairFrame mechanisms to apply to any pair of notes, even ones which are not immediately successive in the surface. The NotePairFrame can then be applied recursively to higher levels of skeletal lines abstracted from the surface.

Forming Reductions and Abstract Lines

Next, for each NotePairFrame that the new NoteFrame is in, the new NoteFrame is added to each of the NoteSequenceFrames which contain the old NoteFrame of the NotePairFrame. If the NotePairFrame's old NoteFrame was not the last member of a NoteSequenceFrame (for example, if several notes had been played in the surface NoteSequenceFrame between the old and the new note), then a new NoteSequenceFrame is created containing both notes of the NotePairFrame. Because there can be several different NoteFrames added as branches from a NoteSequenceFrame, the lists of succeeding NoteSequenceFrames in the node holding the old note of each NoteFrame are updated to reflect each of the possible continuations.

In addition, each of the metric cycle NoteSequenceFrames active in the PhraseFrame check the metric position of the new note relative to their last note to see if the new note is a continuation of their cycle; if so, a new NotePairFrame is made and added to the NoteSequenceFrame. In order to reflect this in the relative Activation of the NoteFrames in the PhraseFrame, the last NoteFrame in each metric cycle NoteSequenceFrame is given a temporary Activation boost so that it will be available to match with the expected next NoteFrame.

As explained in the *Representing Music* section, each of the NoteSequenceFrames can be considered to be an analytic reduction of the important skeleton of the input line. The Emphasis of each of the NoteSequenceFrames determines its place in the ranking of sequences in the PhraseFrame's list. In the system, each feature contributes to this Emphasis in proportion to the length of the RegionFrame over which the feature is constant (see below), scaled by the coefficient (a control variable) for that feature's importance to the analysis. For example, a sequence of notes in an ascending scale will have a single RegionFrame for this feature, and will thus contribute to their containing NoteSequenceFrame's Emphasis. This mechanism constitutes a quantified demonstration of how different features offer competing interpretations of the grouping of notes in the surface along an abstract sequence. The presence of conflicting emphasized NoteSequenceFrames is detected (by comparing the boundaries of strong regions) and registered as a status message of ambiguity. By changing the control variables, different interpretations of the phrase are made available[35].

Labeling Lines

Once each new NotePairFrame has been added to NoteSequenceFrames, the ContourFrames for each sequence update each of the feature's RegionFrames to include the new input. For each feature, when the current RegionFrame's value is the same as the new value, the region is continued; otherwise, a new RegionFrame is constructed (and added to the ContourFrame), and Emphasis is added to the NoteFrame to indicate the perceived accent of the discontinuity. (This makes the NoteFrame more eligible to be matched on a higher level NoteSequenceFrame.) An Unresolvedness value, based on the tension of the new NoteFrame, is assigned to the instance based upon the assumption that the new addition will be the last; if another NoteFrame is later added, the Unresolvedness value is updated to reflect the tension of this later note instead. In addition, new NoteSequenceFrames label their Abruptness by their first note's tension.

Each of the sub-agents (or categories) of the NoteSequenceFrame check if the modified line matches their templates. For example, the GapFill category matches a pitch leap followed by a step in the opposite direction. When there is a match, a RegionFrame is created to contain the feature (or a previous RegionFrame is continued, if one exists for the feature). When no category labels are available for a discontinuous feature, the default Weird label is added (as described in *NoteSequenceFrame Color* section). These unusual features are also sent as status messages to the interface, to indicate that the system has noticed something confusing.

Finding Structure

The representations built by the preceding sections (a PhraseFrame with constituent NoteFrames, NotePairFrames, and NoteSequenceFrames, listed according to Emphasis and labeled in terms of feature values Colors) can be compared against saved representations to determine the structure of the input and its reference to previous material.

Structure within a phrase is found by comparing the elements of the phrase at likely points of repetition. These points are defined by: metric positions which are conventionally used for repetition in the genre; and levels of Activation of the recent instances. There is a convention in this genre of repeating material at successive metric units, on each metric level (as is seen between phrases on higher metric levels in the SoloConventions); thus, new input material is always compared against old material which is in the same position relative to each metric level. For example, at the start of the second measure of a phrase, the start of the first measure and the set of instances with high Activation in the immediately prior material are considered likely points of repetition. This allows for the system to handle both conventional structures, and to be able to use perceived accentuation of an event as a clue to pull out a frame for comparison.

For each such point, a new FeatureMatchEvent is created and an exhaustive comparison of the material at the old and the new points is calculated, listing each type of similarity and difference. Each of the similarities contributes to the Activation of the FeatureMatchEvent. Then, the FeatureMatchEvent is checked against the list of FeatureMatchRegions in the PhraseFrame to see if it is contiguous with any of them: if so, it is added to the region; if not, a new FeatureMatchRegion is created and it is added to that. The Emphasis of the FeatureMatchEvent then increments the Emphasis of the FeatureMatchRegion; the Emphases of the constituent events of a FeatureMatchEvent are also incremented. As with the other instances in the PhraseFrame, a threshold (control variable) then determines which FeatureMatchRegions are sufficiently strong to maintain in memory. The resulting set of FeatureMatchRegions is the structure of the input phrase.

Reference Repetition

Reference to other phrases, either earlier in the song/chorus or in history, is also checked. Activation values are used to enable matches between the new input material and old material based upon similarities for low level features. For each agent, as new instances of the agent's frame are made in the input with particular values, those saved instances for that agent which match these values have their Activation incremented. For example, a new NotePairFrame which is a MAJOR-SC step of two quarter notes at the start of a measure will cause old NotePairFrames with these features to become more active. In particular, this automates the selection of figures which are similarly placed metrically (and thus have the same features in relation to the metric framework) for comparison; the SoloConventions also serve to activate these possible matches. The Activation of these single saved frames spreads to also activate the larger saved figure which contains them. Then all of the active saved frames above a threshold (control variable) are exhaustively compared to the new material in the same fashion as in the above paragraph. A FeatureMatchRegion with a strong Emphasis value thus reflects that the new material is a reference to old material according to the features of the region. A PhraseRelationFrame is then created to contain the comparison of these two phrases.

Placing Phrases in a Solo

A phrase is determined to be complete when its quality of closure (see *PhraseFrames* section) is past a threshold, when several of the NoteSequenceFrames' PredictionEvents (see below) are disappointed (meaning that the lines have ended), or when the SoloConventions indicate that a new PhraseFrame should start; the relative influence of each of these factors is determined by control variables. When a phrase is completed, a new PhraseFrame is created as the currently active phrase, and is placed in the SoloFrame.

A new PhraseRelationFrame is then created, and the successive phrases are compared (see *Comparing Phrases* section). When there are several strong FeatureMatchRegions, but there are also several differences, the label StatementAndVariation is assigned; when the first phrase is an opening phrase and the second is a closing phrase, the CallAndResponse label is assigned. These labels are then sent as status messages to the interface, to indicate that the system has found high-level concepts which it comprehends. The updated SoloFrame is then matched against the SoloConventions; matched features are registered as status messages, again indicating that the listening processes have formed the expected understanding. When these matches do not apply, the system registers that it does not understand (literally, it does not have a category for) the progression between the phrases given their location in the solo.

Expectation

One manifestation of musical understanding is the recognition of implications in input material which generate expectations for specific continuations. The current system is sensitive to three types of expectation: resolution of tension, either harmonically or metrically; continuation of similar features, in particular the tendency for a scale step to be followed by another; extension of a match (the continuation of a pattern played previously). Each agent determines separately whether its features result in expectations of each type.

Each of these types of expectation is handled by projecting expected continuations of NoteSequenceFrames; this is similar to the process used in generation to create new events. The representation of an expectation in an actual future event is called a PredictionEvent; it has a time slot (which by default is simply the time difference between the proceeding events in the NoteSequenceFrame), an Activation (a recency value, which determines how long the expectation continues to hold), an Emphasis (the importance or degree of the expectation), and a label of the feature which it expects to match. The PredictionEvent is linked to the end of the NoteSequenceFrame when the implication is detected. Then, as new events come in to the system, this expectation is either satisfied or disappointed, and registers this result scaled by the Emphasis in a status message to the interface. If the time of the PredictionEvent has passed without the expected event occurring, disappointment is registered; if the feature is matched at the correct time, satisfaction is registered and the PredictionEvent is deleted; if at any time while the PredictionEvent is still active its feature is matched (before or after the expected time), a pointer to the matching event is added to the alternative continuation lists of the NoteSequenceFrame node.

Resolution of tension is implemented by having the agents for harmony and meter increase the Emphasis of this expected next event (using the default values) in the cases where an unstable note has occurred; this results in the system noticing when a line ends abruptly (tones left unresolved, or the rhythm not landing on a strong final beat). In addition, each type of resolution of tension reflects learned conventions for the paths that are usually followed in the genre; the resolution of a scale tone to its successor uses the same abstract mechanism as the resolution of an opening phrase to its usually succeeding closing phrase[36].

Similarity of continuation is implemented by having the NoteSequenceFrames on each level expect that the next event will have the same features as the prior event; this results in having a change or discontinuity noticed by the system as a disappointment of an expectation. For example, each metric cycle NoteSequenceFrame always creates a PredictionEvent for the expected next slot in its cycle; a CrossCycle only makes a PredictionEvent if its Emphasis is high enough. The only possible matching feature in this case is the period, so alternative continuations are those with close but incorrect times. Extension of a match is implemented by checking the relation of the old event (which the current event matched) to its next event, and then expecting a new event that exists in this same relation to the current event.

Color and Affective Results

Describing the character or emotional quality of a piece of music is a difficult endeavor, for it is largely subjective, and imprecisely expressed even in natural language. However, much of understanding music consists of affective responses of suspense, tension, gratification, anxiety, relief, and other emotional terms. [Minsky 89] suggests that in order to form decent theories of music or other types of mental processing, we must acknowledge this emotional character; starting with simple theories of emotion allows us to build systems which use these, even if later we decide that they were incomplete.

In this system, affect is the result of learned knowledge of conventions interacting with processes. The knowledge encoded in the system needs to reflect how certain musical gestures create implications (as discussed in the *Expectations* section) which result in emotional responses. This system implements the theory that the relationship between detected and predicted features constitutes a major element in the affective perception of music. This is roughly based on the implication-realization theory suggested by [Meyer 56] and [Narmour 90], wherein frustrated expectations are a source of affect. In the present system, the above description of expectation provides the quantified mechanism through which affect can be modeled. When disappointment is registered with the interface, this is the sense in which the system represents an emotional response to the music.

In addition to the affects registered from expectation, the present design is extendible to explore definitions of other qualities based upon the colors that are determined in the music. For example: the amount of perceived *forward motion* could be determined by the relation of the melodic tempo to the syncopatedness of the surface; *surprise* could be defined as the correlation of a disappointed expectation with a highly emphasized event; etc. The important point in the current system is to be able to categorize similar types of affects, regardless of how they are labeled. To this end, the Color labels of the levels of the representation serve as descriptions of affective quality. It is not necessary for this system to try to find further adjective descriptions for the difference between a major and a minor scale, or between syncopated and straight rhythms; simply recognizing these and other characters of the music allows for different emotional adjectives to be assigned for specific applications. The present design lays the framework for later work on such definitions.

Effects of Comprehension

Given these types of representations of the musical line itself, specifically of the structures of phrases, parts of phrases, and groups of phrases related to context over time, several phenomena related to comprehension of music can also be modeled. The general idea of chaining musical ideas, or of having musical associations or priming of categories, is represented in this model by spreading Activation of through the network of instances. This allows exploration of high-level ideas about how a solo evolves over time, maintains a sense of direction, or follows a dramatic curve of intensity.

The use of Emphasis and thresholds to determine which instances are maintained as part of the representation reflects the varying degrees of attention and energy which a listener can devote towards remembering music. Adding Emphasis for features such as changes of direction or leaps/discontinuities in the pitch line corresponds to remembering those features which music cognition research have found to be salient in human comprehension.

Phenomena such as the difference between first and second occasions of hearing a piece of music can also be explained given this framework. On the second hearing of a piece, the lower level agents will continue to have their usual local responses to the changes and expectations present in the musical surface, even though the high level structure (a SoloFrame) for the piece is already known. The intuitive notion of using present information in a musical surface to "go back" and change the representation of what was heard previously is also reflected here. As new material comes in and a higher level structure (such as a NoteSequenceFrame or a PhraseRelationFrame) is built which contains the relation between this material to the previous material, the old material is assigned a pointer to this structure and can then determine its role relative to the new material.

3.4.2. Processes of Playing

The purpose of playing is to produce an output solo line that reflects the intentions described by currently active goals. When generating, the system's goals are determined by the user's settings, or the system's current disposition (described by the active SoloFrames). These goals reflect the desired qualitative character of the output.

Improvisation relies heavily upon the use of previously heard material, consisting of both learned figures and tunes, as well as recently played melodic lines [Schuller 68]. Generated output consists largely of repetitions and variations of the most active saved instances of phrases in memory, mixed with new material that is either slightly influenced by or a response to previous material[37]. In addition to the previous phrases, the default metric and harmonic conventions (resolutions to stable choices) are active as default *urges* influencing the player.

The basic ability in the generation model is thus to generate an output phrase based upon the stored representation of a phrase. The power of the representation is demonstrated here, when seen in terms of the knowledge that it provides for generating variations; the system is able to produce variations based upon the structure or qualitative colors of phrases.

The actual generation of an improvisational line is viewed here as an action selection problem. I assume that an improvisation does not follow a simple set of rules, but rather is influenced by a variety of sources concurrently. This type of decision making is well modeled by a spreading activation network which responds to both goals and the environment [Maes 89]. In addition, there need to be components which are able to simply enforce rules, and follow scripts or dependencies on a low or high level. My model defines a hybrid system in which notes are chosen by competition through spreading activation, where the mechanisms controlling the spreading are more rigid components influenced by traditional AI structures of rule-based constraints and script-following fragments. As is the case in listening, the processes described here are intended to model the behavior of playing, rather than simply providing an algorithmic translation of a representation into output notes.

Spreading Activation

When generating a line, the program's goals spread activation through the network of concepts, as do the active ideas in the song environment. As this spreading activation is occurring, the MotorModule plays notes according to the lowest level agents' descriptions; this is similar to the use of pitch and rhythm textures in the DBX system. In other words, the decision of what to do is made by simply checking which notes look best at the time that an action is needed; notes with the highest values when the motor routine is polled are the ones which are considered for output.

To summarize the process of generation: intentions are indicated by SoloFrames, which serve as goals; phrases and parts of phases are activated by matching what is played, by being present in the recent environment, and by matching the desired qualities of the goals; the active instances then suggest low level features of notes to be output by adding them to the MotorModule buffer; the MotorModule buffer plays the winning notes when polled at frequent time intervals[38]. The *actuators* of each agent are its mechanisms for spreading activation and effecting its occurence in the output line.

This use of spreading activation is essentially a way to combine the interests of different musical concepts that the system wants to realize at a given point. However, it also does allow for simple emergent planning behavior [Maes 89]; for example, the concept of GapFill melodic motion requires that leap occur before a resolving step in the opposite direction, and thus spreads activation to the pitches that effect this. However, the most important feature of the use of a spreading activation network here is its capacity to model the phenomenon of chaining musical associations and ideas in the mind of a musician.

Goals and SoloFrames

The goals for the system's generation of an output phrase or solo are determined by combining the contributions of active SoloFrames; the active SoloFrames determine the values for the goals during generation. The SoloFrames thus serve as the high-level descriptors of the program's disposition over the course of the solo. By default, the SoloConventions are active, as is the UserSolo. In addition, any saved SoloFrame instances can be made active by the user to create a specific type of solo; currently this is done manually through the console window, but it will be part of the graphic interface when complete. The active goals during generation determine the nature of the output phrase in terms of intrinsic features (such as its level of motion or syncopation); the output's relation to expected conventional behavior (such as its degree of conformance to default urges) is determined by the relative Activation of the conventions to other influencing material. The *SoloFrames* section describes the types of goals that are thus available, indicating the dispositions of the program when generating.

The use of goals here is based upon the standard use in spreading activation networks for action selection [Maes 94]. The goals increase the Activation of the concepts that produce the qualitative label associated with the goal. Here, this means that saved instances which match the indicated values in the SoloFrames have their Activation incremented for each feature that is matched. This activates both entire PhraseFrames, as well as those constituents of a phrase (NoteSequenceFrames, RegionFrames, etc.) which cause the indicated feature. For example, a goal Color of syncopation will activate PhraseFrames which are generally syncopated, and within these will further emphasize the NoteSequenceFrames that have syncopated regions.

Conventions

Conventional behavior within a genre can be viewed as default behavior that meets the expectations of the listeners; in this sense, it can be contrasted with either making mistakes, or creating novel acceptable solutions to situations. The simple low level conventions which we represent here reflect the basic structures of the metric and harmonic frameworks: by convention, stable elements are preferred, and unstable elements resolve to stable elements. For example, the meter and harmony resolution conventions generally spread activation to output notes according to the PitchSeries they are in (more Activation for lowest harmonic tension) and to output metric positions according to the highest metric level they are in (more Activation to higher metric levels). In addition, these conventions can be effected through active saved instances of NoteSequenceFrames (traversing scales) which resolve unstable pitches.

In addition to these surface conventions, structural conventions are represented through the SoloConventions (described in *SoloFrames* section). The SoloConventions are always active by default; these indicate likely places for phrases to start, and suggest that certain phrases be variations of other phrases. The studied genre has established high level expectations for deviations of low level conventions: by convention, syncopations and chromaticisms are frequently emphasized in the course of a solo, and are reflected in the SoloConventions. These two seemingly conflicting influences are combined by giving the low level conventions a small constant Activation boost, to encourage default choices to be traditional stable elements, while the high level SoloConventions periodically provide a large Activation boost to unstable elements, creating frequent alternations of Color deviations with the stable elements.

Environment/Musical Context

The other major factor influencing the output is the environment, which here is the musical context. The most important influence here are the currently active PhraseFrames; the melodic lines of the tune and of previous solos remain active and thus affect the output, either through vague influence or by direct reference. In addition, the environment as defined by the current position in the song is used to influence the output: the metric agent temporarily increases the activity of saved instances which occurred over the same metric position, and the harmony agent temporarily increases the activity of saved instances which occurred over the same harmony (current and next chord). These increases are proportional to the degree of the match; this allows for an exact metric position match (same beat in the solo) to be most activated, while a similar one (same measure, or same beat relative to measure) is less activated, and for an exact harmony match (same absolute chords) to be most activated, while a similar one (same current chord with different next chord, same relative relation between chords) is less activated. This mechanism allows for the effect of having a set of chord changes (such as a turnaround at the end of a solo) remind the player of common ways of playing over that stretch of measures [Berliner 94]; for example, Armstrong frequently plays an eighth note arpeggio descending from an offbeat high 'F' in the final phrase of solos.

Feedback

Another aspect of the musical context is feedback from the generation. As the system generates a melodic line, it listens to this line as well; the constructed representation by the listening processes of what was played allows for feedback within the system. This feedback serves two purposes: first, it reflects the degree to which the actual output realized the intentions; second, it acts as influencing environment information by suggesting paths for the continuation of the solo. The former purpose is similar to Rowe's use of a critic [Rowe 91]; in the current system, this information is not yet used to change the output, but is useful in monitoring its success. The latter purpose, however, is a major force in determining the course of the output.

As the analyzed line is matched to previous saved instances of the representation, these old instances (and their containing phrases) become active and are thus considered in creating the output line. It is through this mechanism that the system can simulate having a chain of musical associations: when it plays something that "reminds" it of something it knows previously (such as a pair of intervals), the old material is then available to influence the output. In order to avoid having one string of saved instances completely dominate the output by gaining increasing activation, the Fatigue threshold (a control variable) indicates a maximum Activation level; if an instance goes beyond this level, its Activation is reset to zero, to reflect that the instance has been overused. While this is an oversimplified solution to the problem, it is based upon a plausible cognitive mechanism for avoiding domination by one module, as found in other systems motivated by ethology [Blumberg 94; Tyrrell 93].

Generating Variations

There are many possible degrees of variation, which can create a functional substitute (restatement) of the original, an elaboration of it, a noticeably influenced new phrase, or a vaguely influenced new phrase. The original phrase is represented in terms of the PhraseFrame structures described above; thus, the imitation of this phrase can use any of the features of this representation to guide the construction of a new phrase. While this can be done by literally repeating sections of the melodic surface, it can also be accomplished by repeating other salient features, such as the reduced skeleton, the degree of syncopation, the presence of a CrossCycle, or the type of elaboration of the original skeleton (placed on a new skeleton).

When the goals/SoloFrames indicate variations of specific phrases, those aspects of the original phrase (such as a specific NoteSequenceFrame level) which correspond to the desired colors are maintained, while the other levels are allowed to change. That is, the desired features are given high Activation, while the other features are not; this allows for competing phrases and default urges to fill in new features to complete the phrase partially specified by the desired features. For example, if the system intends to maintain the overall skeletal pitch sequence and sense of closure of a phrase, it can effectively borrow a CrossCycle or a set of elaborating pitches from another active phrase; the pitch skeleton can be projected onto the CrossCycle NoteSequenceFrame, and the elaborating pitches can fill in the lower level or surface NoteSequenceFrame. This allows for the combination of a desired color, such as a rhythmic feel, with a desired pitch reduction; this is realized through spreading activation to desired instances and combining evidence on their votes for the output.

While this set of processes appears to simply combine previously known patterns, it actually can generate a wide range of possibilities through the combination of influences. This system does not simply concatenate sequences; rather, each active sequence combines its effect with the others to determine a result. For example, the pitches of one sequence can suggest the melodic direction for the output while the pitches of another suggest the melodic tension. Essentially, previous phrases serve as general melodic feature suppliers; rather than having every type of feature in a separate agent, clusters of these features as realized in learned instances are used instead[39].

The above sections explain the mechanisms through which a set of instances become active to contribute to generation. The actual process used to suggest new output notes is the same as that used to generate predictions when listening: each of the NoteSequenceFrames that are active uses its actuators to project its expected continuation by adding the future note to the MotorModule's buffer. This expected continuation is simply the next node in the sequence ordinarily, except when there is no next node, in which case the prediction is of a continuation of previous motion (such as the same scale step, direction, and time interval features as in the previous NotePairFrame). A NoteSequenceFrame spreads activation along its chain of nodes to always increment the Activation of the next node in the sequence when it should occur, and the previous node in order to bring about its own realization; thus each node prepares its successor, and requires its predecessor. The basic algorithm of following pointers through the list of nodes which make a sequence is the obvious way to playback a sequence. The interesting aspects here are: that several active influences are combined to form the output; that this occurs on each of the levels of the representation (e.g., the surface and skeletal levels); that the node at each point influences the output separately in terms of each of its features; and that the spreading activation allows for the most active nodes to be realized and resolved (approached and concluded) automatically by appropriate adjacent nodes.

MotorModule

The actual output notes come from the MotorModule. This module uses of a buffer which has slots for each smallest metric subdivision over a variable amount of time into the future; the current system's furthest time slot is one measure in the future. Each time slot then has a list of possible notes that can be played at that time step, including velocity (a velocity of zero indicates a rest) and Activation values. An element of the representation (such as a saved instance of a NoteSequenceFrame) "votes" on an output note by adding the desired note to the desired time slots, and incrementing its Activation.

The MotorModule is then polled at each smallest metric subdivision to determine if a note should be played. It checks the current time slot to see if any note's Activation is above a threshold (a control variable), and if so plays the note with the highest Activation. Then the buffer is reset and current note information is erased; this ensures that new events will only be suggested if they continue to be valid given the choice that was just made. The MotorModule is thus the only place where exclusive competition takes place. Everywhere else, Activation is simply summed as combination of influences, which eventually suggest output options to the MotorModule to decide between.

Planning Ahead

We would like for the system to be able to plan ahead, to create present trajectories that will land it in desired positions at future times. In the sections above, the spreading activation network's capabilities for planning of desired qualitative goals are only minimally used: spreading activation here is essentially combination of evidence, such that desired goals are achieved through activating instances that will realize them and are active given the environment. The type of planning required here for surface trajectories, however, is fundamentally different from that referred to by [Maes 94] in that the absolute time of the goal of the trajectory is important, not simply the order in which a sequence of actions is executed to get to this goal. For example, if we want the system play the tonic on the approaching downbeat and to approach the tonic by stepwise motion, each of the steps leading to the tonic is itself a subgoal which should happen at a specific time. Simply finding a path through spreading activation from the current note to the goal note is not acceptable, because the goal note needs to be played a specific time.

A simplified design of the generation module approaches this problem by using learned paths to goal tones. Just as improvising musicians have a stock of known figures that are played in certain situations in a chorus form, the system is effectively given a set of learned paths that can cause it to arrive at its goal on time. This is implemented by considering the steps around the goal tone to be goal tones themselves, given a portion of the goal tone's activation; this process continues recursively to suggest approaches to the goal. This allows for the system to freely generate a melodic line, and as it approaches a goal to be increasingly influenced to get on one of the paths towards the goal.

While this model is usually successful at realizing goal tones in its lines, it is overly simplified: the only types of goals considered are tonic notes played on downbeats, and these are only approached by stepwise motion. A more complete design and implementation, using the entire representation described above, is required to allow for flexible and consistent planning of trajectories; this section of the system design has not been completed. The intention is to capitalize upon the representation's use of several NoteSequenceFrame's, each of which projects its own future goal for a next step. This is meant to correspond intuitively to the different levels of planning for the future that the improviser maintains in his mind: the next surface note, the next middle level metric pulse on a phrase skeleton, the next major downbeat, etc. The addition will be to use these future goals to influence current decisions. From each of these goal tones, the interpolated paths based upon the PitchSeries can be placed as sub-goals in the MotorModule's buffer, which then themselves can spread activation back to continue the paths. Awkward or sudden leaps to goals would be avoided because the output buffer would cause other nearby tones to occur before the goal tone.

Generation Behavior Effects

This section briefly summarizes several phenomena related to our creation of music are modeled by these generation processes. As described for listening, the general idea of chaining musical ideas is represented in this model by spreading activation through thenetwork of instances. When used for generation, the resulting improvisation is ideally a series of cascading choices that reveals the intentions of the player, opportunistically reacting to the musical context; an unfolding gestural idea is realized over a changing background of the song environment.

Instances that are active in the model are analogous to having ideas "in mind" for a human. Self-criticism is available through the feedback mechanism. Thinking ahead or imagining possible continuations of the improvisation are literally reflected in the system's projection of NoteSequenceFrame predictions into the future. There is no provision in this model for strict logical reasoning or deep search involving much backtracking, a restriction which is in keeping with the human model of most musical processing. However, some pseudo-logical reasoning is implicit, and necessary for good results, in the spreading of activations over chains of ideas, which constitutes a type of prediction or planning.

4. Evaluation

The concrete goal which I committed to completing during the master's timeline was the design of a listening module that performs an analysis of input solos which reflects characteristics in multiple representations. This goal has been achieved. The emphasis of the program development has been on the listening functionality, because this most clearly demonstrates the system's musical sensibilities in terms of what features it knows about. I hope that this will suffice for demonstrating the efficacy of my perspective, of how my approach may be used to create an intelligent system.

The *Processing Music* section provides a thorough evaluation of the representations presented here through a demonstration of their use for analysis and generation tasks. This section briefly summarizes these contributions in relation to the problems of previous systems outlined in the *Introduction.*

The major frustration experienced with previous systems is their neglect of intuitive representations of music. The present system is based on a working theory of music, and has explicit representations for the features and structures described in the *Understanding Music* section. The best demonstration of the relevance of these features and structures is the *Example Comparisons* section, wherein the system's representations allow for intuitive comparisons to be made. The general approach presented in this document is extendible to handle other features and structures which may be considered to be of importance. Adding to the current representations will develop the system's musical common sense.

There are several broad advantages to the current system. In particular, the current representations allow for a definition of musical context, development, and the relative importance or salience of features. Similarity of material is recognized; repetition of material can be produced, and also varied over a solo. New input is recognized in terms of its relation to conventional material. Summaries and abstractions of surface lines are available through the emphasized components of PhraseFrames.

The benefits of my approach are evident in comparison with the DBX and DrumBoy systems. Notes in the current system are labeled in terms of the structures (e.g., reductions, groups, repetitions, etc.) they are part of; thus, comparisons of phrases respect the important features, as do generations of variations. The relation of material (input or output) to conventional expected material is recognized; deviations and unusual features are noticed and categorized or labeled as confusing (Weird). The analysis recognizes the absolute color characteristics of phrases, enabling the system to judge its own output both for its intrinsic features and its submission to the conventions of the genre. The SoloFrame construct describes development over a chorus (a longer time period than either of the prior systems) in terms of phrase relations and intrinsic features. The system is able to produce variations based upon any element of a phrase's structure, in combination with other influencing material or desired goals; this enables sophisticated response to input material, and is not restricted to a database of known patterns.

These abilities are the result of the fundamental approach which motivates this system. Much work remains to be done, both in implementation of a working system, and in extensions of the design to incorporate further features and other improvements. The important point is that the explicit representation of common sense musical concepts in an agent-based architecture allows for the construction of systems that are able to behave more intelligently.

4.1. Implementation Status

All the components of the design listed in the *Building a More Intelligent Music System* section are quantified sufficiently to be coded, but not all have been completed due to shortage of time[40]. The current implementation has completed: the utilities (scheduler, handling MIDI); the framework for the containment and management of agents/K-lines (including message passing protocols, and the default base classes); the frame classes described in the *Representing Music* section; the listening modules for most of the functions described in the *Processes of Listening* section (including finding abstracted lines, patterns, color labels, similarities, and matches); and a test spreading activation network for generation using a simplified representation.

The main portion of the code remaining to be completed is the mechanics of spreading activation for generation according to the current version of the design described in the *Processes of Playing* section. (The test program for the generation processes is based on the same ideas, but does not use the full representation in the current system.) Some of the listening modules need to be updated from an earlier version to accord with the class names used here; also, better pruning mechanisms need to be added to keep the number of saved representations smaller than it currently is. In addition, the graphic interface has not been completed. Finally, the parts are not all connected yet into a simple demonstration application (see *Appendix 1*).

4.2. Limitations

There are three major categories of limitations of the current approach and system as described above: 1) inefficient aspects of current operation; 2) absence of learning mechanisms; 3) generalization of individual differences; 4) incomplete affect and world knowledge. The largest technical problem in the system is the management of spreading Activation and changing Emphasis values. The major inefficiency of the analysis is the absence of better pruning mechanisms for keeping the number of saved structures to a minimum. Currently, the PhraseFrames simply have sorted lists into which all of the smaller instances are placed in order of Emphasis; the number of instances saved can only be decreased by changing the size of these lists. The difficulty is in knowing which of the structures will become necessary again in the future. Since this is hard to determine, I currently keep most of the structures around unnecessarily. It would be better to find a smooth way to recompute them automatically when needed. Ideally, the system would change its own control variables to automatically regulate the amount of Emphasis which instances received according to the current context. In general, the listening processes need to be abstracted from their function in analyzing an input stream to be applicable at any time to any material indicated by the user through an interface, or to respond to requests by the generation processes. The largest inefficiency in the generator is the tendency for the spreading activation network to get caught in "ruts" of repetitive behavior. Meta-level components should be added that monitor the behavior as a whole to avoid these situations, in order to ensure that conventions are respected to the right degree and that the output is produced when needed. These components would be similar in functionality to the "B-brain" agents described by [Minsky 86].

The absence of any learning mechanisms is a larger problem. The system currently processes the phrases that are played in during its execution, and then "forgets" these when the program is quit. A simple type of learning that should be added is the ability to add phrases automatically to the database of saved instances, rather than doing this manually. More importantly, higher level learning to categorize styles or qualitative features of input should be experimented with given the framework provided here. Further, the feedback provided by the listening processes when generating should be used to correct output that fails to achieve the goals; ideally, this correction would change thresholds and activation values such that on future occasions, the output would be more accurate.

The above description of the system generalizes information about musical understanding, while in fact human listeners do not really share a single common musical understanding. Listening is knowledge intensive and personal: your experience determines what you expect, what you actively look for, multiple ways of interpreting material, and ways for handling structures or types of structures that you haven't seen before. Different listeners will derive different structures; however, if we assume that listeners begin with common mental capabilities, then we might suggest that they build similar structures, some more elaborated than others. Introspection is, unfortunately, insufficient to determine how we understand music; the processes and representations suggested here are based upon observed behavior and the knowledge which we know contributes to this behavior.

The largest general problem which I have encountered results from the fact that I am attempting to separate one aspect of intelligence from a human system which apparently does not respect modular boundaries of locations for specific abilities itself. In other words, there are unavoidable deficiencies in the overall success of my approach towards musical intelligence due to the unsolved questions of intelligence in general. These gaps of knowledge range from processing issues, such as the fact that music processing uses many mechanisms which arose in the brain for other purposes (ranging from general imaginative capacities to inferencing methods, the complexity of which are far beyond the scope of this research), to qualitative issues, such as the fact that my system does not know what "Autumn Leaves" actually are, nor does it have a developed representation of melancholy.

Many questions of affect in particular are out of reach due to this situation, and due to the fact that we don't have good theories of how emotions work. The affect knowledge about music contained in the system is mainly limited to the Color labels, references to other phrases, and the status messages indicating the progress of the listening processes. Our long term goal, of course, is for the system to simply understand what we think sounds "good" or "cool". While these are not possible now, the current system provides much information about the structures and relationships between parts of music which are components of our affective concepts about music. I hope that my work will contribute towards enabling further understanding of the musical processes related to emotion.

5. Conclusion

Music may appear to be a mysterious "bug", but it can still be subject to formalization and modeling. The attempts to understand natural language in the past half century have yielded a range of usable specialized systems and, more importantly, have offered a clear evaluation of where our intuitive theories of language are deficient as explanations. The same attention must be paid to music if we are to increase our understanding of musical intelligence, a necessary step towards building better computer music systems.

This work was motivated by frustrations with previous systems, in particular their inability to notice salient features about music. Previous systems have suffered from an absence of representations for common sense knowledge about music. This thesis proposes that we must enumerate the concepts which constitute our knowledge if we would like for our systems to share them. This is a huge problem; I have approached it from a perspective influenced by symbolic artificial intelligence, and created an architecture for the representation of musical knowledge in a computer system.

The architecture proposed by this thesis emphasizes explicit representation of structures humans notice when listening and intend to create when playing. The representation models how these concepts may be recognized and produced. Each concept is embedded in a network of other musical concepts, so that the meaning and consequences of concepts and instances are defined by their relations to each other. The system described in this document explicitly represents several important musical features missing from other systems (such as our DBX system), including multiple interpretations of reductions, groupings, and colors (such as types of syncopation) of regions. More important than the specific system presented here is the general approach to building better music systems by quantifying aspects of musical intelligence. I suggest that it is through such explicit enumeration and quantification of musical concepts, which can represent different and conflicting perspectives, that progress can be made towards understanding musical intelligence. The next step is the categorization of high level qualities, such as the affective meaning of music; the current work provides the framework for this endeavor.

My intention is to set a precedent and offer a framework for continued work in this direction, a direction that emphasizes the fact that musically intelligent systems must be extremely knowledge intensive, and also have flexible representations and control structures that correspond (roughly) to the vast multiplicity of methods, ideas, and devices which humans utilize when listening to or playing music. Systems built according to these principles will allow for further exploration into questions of musical intuition and aesthetics. Using higher level musical knowledge as the atomic units for standard learning algorithms will allow development and discovery of musical concepts which are rooted in our understanding of music. Thus, the large scale of the enterprise here is to lead the way into the development of systems which will allow us to directly explore human music cognition. In addition to being a scientific study, this is an opportunity for the design of multitudes of applications which take advantage of these quantified aspects of musical understanding to entertain, teach, communicate, challenge, and otherwise aid us in self development and expression.

Appendices

1. Demonstration and Applications

The interface for a demonstration version of the program is currently under development. This section briefly describes the design for the demonstration, and possible future applications.

The goal of the demonstration is to allow users to explore the workings of the model. The main functions are the analysis of an input phrase and the generation of an output variation of this phrase. The user can start and stop the demon which outputs sequence-like accompaniment instruments' lines, playing over the selected song's chorus form. As the accompaniment plays, the user inputs a solo (from a MIDI device) and the system analyzes it in real time. The screen display indicates the progression of the analysis: notes are drawn as they come in, structures are drawn as they are discovered, and status messages are printed in the console window. The screen display depicts the surface and structure of phrases in the form of the diagrams of the *Example Comparisons* and *Levels of NoteSequenceFrames* sections, using color to indicate the relative Emphasis and Activation of instances; the Color features of PhraseFrames are listed at the bottom of the window. The user can print (in the console) a detailed text description of an instance or a link by selecting it in the diagram. The top of the window lists the control variables as sliders which the user can change to affect the program's analysis; a future version will reimplement the current listening functions to allow them to automatically update the previous analysis based upon these changes instead of simply performing their next analysis with the new values.

The user can select an analyzed phrase to be output by the system. Variations of the original phrase can be generated by changing (through the interface) the relative Activation of its structures. The selected song is also displayed on the screen as a grid of rectangles representing the measures of the chorus form, each of which contains the active SoloConventions and the UserSolo. By selecting a SoloFrame, the user can change its values, including its Activation. Further output variations can result from changing the desired features in the UserSolo, and the Activation level of other influencing phrases. A future version should allow for the database of saved instances to be easily available for examination; the user could then view the system's comparison of two phrases by selecting them.

Once these basic elements are in place, there are a variety of further applications which could be constructed on top of them, both to demonstrate the system, and to create interactive systems. A future goal is to create a simple SoloConvention script indicating that the program should automatically create variations on input phrases, in order for it to "trade fours" with a user. Alternately, the system could be used as a new type of instrument (similar to DBX), by attaching input controllers (using voice, hand gestures, cues from videos or games, etc.) to steer the output. This system would allow an improvement over the current selection of "computer instruments" available because the system understands the important features of the music that are necessary to reflect and express the input gesture: salient aspects of the control gestures can thus be mapped onto salient aspects of the music using the above representation. Finally, the current system architecture could be used as the basis for a learning system to quantify new musical concepts. For example, if we want to tell a human soloist to "open up more", we do this by giving examples and describing symbolically what this means in the terms which the human already knows. Ideally, we would like to be able to create a new concept in the computer system which reflects this new concept as well, either by explicitly defining the concept in terms of previous concepts, or by running a learning algorithm which uses the previous concepts as part of the signal that is being categorized.

In addition to the above applications, there are many broad future goals that an intelligent system should be able to achieve. A generative system which understands the implications of musical material should know how much of a theme needs to be repeated in order for it to be recognized by different listeners, and how much time it takes to process this match. It should be able to set up expectations for particular events, and then manipulate these expectations in order to create affective results. A listening system should be able to recognize when structures have been suggested and not satisfied; for instance, it should register disappointment and confusion when a musical problem is not solved, or an idea is not completed. It should be able to use its fundamental musical knowledge to learn categories of music, both in terms of styles and affective qualities. The time is approaching when we will have systems that are able to identify a solo as being "bluesy", or a melodic fragment as being particularly "tender".

2. Iambic Meter

[Povel and Okkerman 81] conducted a series of experiments in which subjects were presented with a series of identical tones (equitone pulses of equal loudness). The subjects consistently tended to group these into pairs even though no such distinction existed in the actual acoustic signal[41]. The specific claim suggested by some researchers in interpreting this data is that some interruption of the amplitude processing (in human perception) has occurred when a second note occurs quickly following the first, and thus the first is heard to be weaker than the second, whereas if the time interval is slightly different, the second note's amplitude may be considered to contribute to the perceived amplitude of the first, making the first sound stronger. However, these claims are not clearly supported by the available evidence; in particular, it is not clear how musical background and training, or even gestalt psychological properties of grouping or recency/primacy effects influence these results. While the theory of interrupted amplitude processing is consistent with these results, it is also possible that the results are the product of a higher-level cognitive operation, such as a categorization, labeling, or memory error. In other words, it is entirely possible that the amplitude of the first tone is fully processed, but that this processed information is simply mismanaged cognitively by the subject. This confusion is an inevitable result of attempting to develop a theory of an operating mechanism of perception while only testing the functionality of perception: having done experiments to determine what humans perceive, the question of how this perception is accomplished is still underconstrained. Regardless, what is clear is that on either a perceptual or cognitive level, perhaps innately or perhaps due to training, humans do form groups corresponding to the levels of music theory's metric hierarchy.

This in an intuitively appealing theory, because it seems to relate to many human functions such as the iambic alternation and periodicity found in speech, or in walking, or even a human heartbeat. The results presented by Povel and Okkerman are interesting for the insight that they give us with respect to human auditory perception of musical rhythm, as is discussed in their paper. The interest of the present thesis is to describe a theory of processes governing the understanding of this music. Towards this end, I adopt this organizational principle of hierarchical paired elements as a modular process that the system is capable of performing.

References

Agre and Chapman (1988). "What are Plans for?". A.I. Memo #1050. MIT.

Ames, C. and Domino, M. (1992). "Cybernetic Composer: An Overview", in *Understanding Music with AI.* MIT Press, Cambridge.

Armstrong, L. (1936). *Swing That Music.* Da Capo Press, New York.

Baker, D. *Jazz Improvisation: A Comprehensive Method for All Musicians.* Alfred Publishing, Van Nuys, California.

Berliner, P. (1994). *Thinking in Jazz.* University of Chicago Press, Chicago.

Bharucha, J. (1994). "Tonality and Expectation", in *Musical Perceptions.* Oxford University Press, New York.

Biles, J. (1994). "GenJam: A Genetic Algorithm for Generating Jazz Solos", in *International Computer Music Conference 1994 Proceedings,* ICMA, California.

Berliner, P. (1994). *Thinking in Jazz.* University of Chicago Press, Chicago.

Blumberg, B. (1994). "Action-Selection in Hamsterdam: Lessons from Ethology", in *Proceedings of the Third International Conference on the Simulation of Adaptive Behavior*, Brighton.

Clynes, M. (1978). *Sentics: The Touch of Emotions.* Anchor Press, Garden City.

Clynes, M. (1982). *Music Mind and Brain: The Neuropsychology of Music.* Plenum, New York.

Collier, J. (1983). *Louis Armstrong, an American Genius.* Oxford University Press, New York.

Cope, D. (1991). *Computers and Musical Style.* A-R Editions, Madison WI.

Desain, P. and Honing, H. (1992). *Music, Mind, and Machine: Studies in Computer Music, Music Cognition, and Artificial Intelligence.* Thesis Publishers, Amsterdam.

Dowling, W. (1994). "Melodic Contour in Hearing and Remembering Melodies", in *Musical Perceptions.* Oxford University Press, New York.

Ebcioglu, D. (1992). "An Expert System for Harmonizing Chorales in the Style of J.S. Bach", in *Understanding Music with AI.* MIT Press, Cambridge.

Egozy, E. (1995). *Deriving Musical Control Features from a Real-Time Timbre Analysis of the Clarinet.* Master's Thesis, MIT Electrical Engineering and Computer Science.

Ferguson, D. (1960). *Music as Metaphor.* University of Minnesota Press, Minneapolis.

Feulner, J. (1993). "Neural Networks that Learn and Reproduce Various Styles of Harmonization", in *International Computer Music Conference 1993 Proceedings,* ICMA, California.

Fraser, R. (1993). "Computer Perception of Phrase Structure", in *Contemporary Music Review: Music and the Cognitive Sciences*, V9, Harwood Academic Publishers.

Handel, S. (1989). *Listening: An Introduction to the Perception of Auditory Events.* MIT Press, Cambridge.

Hiraga, Y. (1993). "A Computational Model of Music Cognition Based on Interacting Primitive Agents", in *International Computer Music Conference 1993 Proceedings,* ICMA, California.

Horner, A. and Goldberg, D. (1991). "Genetic Algorithms and Computer Assisted Music Composition", in *Proceedings of the Fourh International Conference on Genetic Algorithms.*

Horowitz, D. (1992). "Intelligent Accompaniment", in *International Computer Music Conference and Festival at Delphi.* Greece.

Horowitz, D. (1994a). "Generating Rhythms with Genetic Algorithms", in *International Computer Music Conference 1994 Proceedings,* ICMA, California.

Horowitz, D. (1994b). "A Hybrid Approach to Scaling Action Selection". Unpublished paper, available from the author; MIT Media Lab.

Horowitz, D. (1994c). "Musical Improvisation with Agents". Unpublished paper, available from the author; MIT Media Lab.

Horowitz, D. (1995). "Representing Musical Knowledge in a Jazz Improvisation System", in IJCAI Workshop on Aritificial Intelligence and Music, Montreal.

Jackendoff, R. (1987). *Consciousness and the Computational Mind.* MIT Press, Cambridge.

Johnson-Laird (1991). "Jazz Improvisation: A Theory at the Computational Level", in *Representing Musical Structure.* Academic, London.

Kernfeld, B. (1981). *Adderley, Coltrane, and Davis at the Twilight of Bebop: The Search for Melodic Coherence.* Doctoral Dissertation, Cornell University.

Krumhansl, C. (1990). *Cognitive Foundations of Musical Pitch.* Oxford University Press, New York.

Laske, O. (1992). "In Search of a Generative Grammar for Music", in *Machine Models of Music.* MIT Press, Cambridge.

Lenat, D. and Guha, R. (1990). *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project.* Addison-Wesley, Mass.

Lerdahl, F. and Jackendoff, R. (1983). *A Generative Theory of Tonal Music.* MIT Press, Cambridge.

Lerdahl, F. (1985). "Cognitive Constraints on Compositional Systems", in *Generative Processes in Music.* Oxford University Press, New York.

Levitt, D. (1981). *A Melody Description System for Jazz Improvisation.* Masters Thesis, MIT, Cambridge.

Levitt, D. (1985). *A Representation for Musical Dialects.* Doctoral Dissertation, MIT, Cambridge.

Lewis, G. (1994). "Interacting with Latter-Day Automata". Workshop at *International Computer Music Conference 1994,* Aarhus, Denmark.

Lidov, D. (1975). *On Musical Phrase.* Groupe de Recherches en Semiologie Musicale, Universite de Montreal.

Machover, T. (1992) *Hyperinstruments: A Progress Report.* Available from the Media Lab, Massachusetts Institute of Technology.

Maes, P. (1989). "How to do the Right Thing", in *Connection Science Journal* Vol. 1, No. 3.

Maes, P. (1990). "Situated Agents Can Have Goals", in *Robotics and Autonomous Systems* 6: 49-70.

Maes, P. (1994). "Modeling Adaptive Autonomous Agents", in *Artifical Life Journal* Vol. 1, No. 1 and 2. MIT Press, Cambridge.

Matsumoto, F. (1993). *Using Simple Controls to Manipulate Complex Objects: Applications to the Drum-Boy Interactive Percussion System,* Master's Thesis, MIT Media Lab.

Mehegan, J. (1959-1965). *Jazz Improvisation,* in four volumes. Watson-Guptill Publications, New York.

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

Minsky, M. (1975). "A Framework for Representing Knowledge", in*Readings in Knowledge Representation.* Morgan Kaufman, Los Altos85, pp. 245-262.

Minsky, M. (1980). "K-Lines: A Theory of Memory", in *Cognitive Science* 4: 117-133.

Minsky, M. (1986). *The Society of Mind.* Simon and Schuster, New York.

Minsky, M. (1989). "Music, Mind, and Meaning", in *The Music Machine.* MIT Press, Cambridge.

Minsky, M. and Laske, O. (1992). "A Conversation with Marvin Minsky", in *Understanding Music with AI.* MIT Press, Cambridge.

Narmour, E. (1990). *The Analysis and Cognition of Basic Melodic Structures: The Implication-Realization Model.* University of Chicago Press,

Owens, T. (1974). *Charlie Parker: Techniques of Improvisation.* Doctoral Dissertation, University of California, Los Angelos.

Pennycook et al. (1993). "Toward a Computer Model of a Jazz Improvisor", in*International Computer Music Conference 1993 Proceedings,* ICMA, California.

Pierrehumbert, J. and Hirschberg, J. (1990). "The Meaning of Intonational Contours in the Interpretation of Discourse", in *Intentions in Communication.* MIT Press, Cambridge.

Povel, D. and Okkerman, H. (1981). "Accents in Equitone Sequences", in *Perception and Psychophysics,* vol. 30.

Pressing, J. (1985). "Experimental Research into Musical Generative Ability", in *Generative Processes in Music.* Oxford University Press, New York.

Pribham (1982). "Brain Mechanism in Music", in *Music Mind and Brain.* Plenum, New York.

Ramalho, G. and Ganascia, J. (1994). "Simulating Creativity in Jazz Performance", in *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA.

Reeves, S. (1989). *Creative Jazz Improvisation.* Prentice Hall, New Jersey.

Riecken, R. (1992). "Wolfgang: A System Using Emotion Potentials to Manage Musical Design", in *Understanding Music with AI.* MIT Press, Cambridge.

Rigopulos, A. (1994). *Growing Music From Seeds: Parametric Generation and Control of Seed-Based Music for Interactive Composition and Performance* Master's Thesis, MIT Media Lab.

Roads, C. (1980). "Artificial Intelligence and Music", in *Computer Music Journal* 4(2). MIT Press, Cambridge.

Roads, C. (1985). "Improvising with George Lewis", in *Composers and the Computer.* William Kaufmann, Los Altos, California.

Rosenthal, D. (1992). *Machine Rhythm: Computer Emulation of Human Rhythm Perception.* Doctoral Dissertation, MIT Media Lab.

Rowe, R. (1991) *Machine Listening and Composing: Making Sense of Music with Cooperating Real-Time Agents.* Doctoral Dissertation, MIT Media Lab.

Rowe, R. (1993). *Interactive Music Systems.* MIT Press, Cambridge.

Schank, R. (1975). *Conceptual Information Processing.* North Holland.

Schank, R and Abelson, R. (1977). *Scripts, Plans, Goals, and Understanding.* Erlbaum, Hillsdale, New Jersey.

Schank, R. and Riesbeck (1981). *Inside Computer Understanding.* Lawrence Erlbaum, New Jersey.

Scherer, K. et al. (1984). "Vocal Cues to Speaker Affect: Testing Two Models", in *Journal of the Acoustical Society of America* Vol. 76, No. 5, November.

Schiff, R. (1961). *Louis Armstrong, A Jazz Master.* MCA Music Publishing, MCA Inc.

Schottstaedt, W. (1989). "Automatic Counterpoint", in *Current Directions in Computer Music Research.* MIT Press, Cambridge.

Schuller, G. (1968). *Early Jazz: Its Roots and Musical Development.* Oxford University Press, New York.

Smith, G. (1983). *Homer, Gregory, and Bill Evans? The Theory of Formulaic Composition in the Context of Jazz Piano Improvisation.* Doctoral Dissertation, Harvard University.

Stewart, M. (1973). *Structural Development in the Jazz Improvisational Technique of Clifford Brown.* Doctoral Dissertation, University of Michigan.

Sudnow, D. (1978). *Ways of the Hand: The Organization of Improvised Conduct.* Harper and Row, New York.

Sundberg (1982). "Speech, Song, and Emotions", in *Music Mind and Brain.* Plenum, New York.

Todd, P. and Loy, D eds. (1991). *Music and Connectionism.* MIT Press, Cambridge.

Tyrrell (1993). "The Use of Hierarchies for Action Selection", in *Proceedings of the Second International Conference on Adaptive Behavior.* MIT Press, Cambridge.

West, Howell, Cross (1991). "Musical Structure and Knowledge Representation", in *Representing Musical Structure.* Academic, London.

Widmer, G. (1992). "A Knowledge Intensive Approach to Machine Learning in Tonal Music", in *Understanding Music with AI.* MIT Press, Cambridge.

Widmer, G. (1994). "Learning Expression at Multiple Structural Levels", in *International Computer Music Conference 1994 Proceedings,* ICMA, California.

Williams, J. (1982). *Themes Composed by Jazz Musicians of the Bebop Era.* Doctoral Dissertation, Indiana University.

Zicarelli, D. (1987). "M and Jam Factory", in *Computer Music Journal* 11(4): 13-29.

[1]This statement is supported by the fact that music seems to have no evolutionary origin; it cannot be traced backwards in our ancestry as a budding or necessary functionality [Minsky and Laske 92; Jackendoff 87].

[2]The common types of generative systems are as follows: systems which take numerical algorithms (e.g., from fractal or chaos theory) and assign the output to musical sounds; systems which produce idiosyncratic musical behavior (sometimes dubiously dubbed "free jazz"), usually in an interactive setting wherein the human performers and the computer process each other's notes; systems which follow precomposed scores with a limited number of options at certain points and ways of altering the realization of the score; and systems which try to sound like popular music, usually through "sequence lookup" (the music that is created by the system consists of the concatenation of prewritten sections of music). Most of these systems can be described as parametric systems, containing a generative module whose output music is altered by the settings of parameters, and an analysis module which produces sets of parameters describing input music.

[3]It is theoretically possible that numerical learning algorithms applied to input streams of pitches will at some point in the future manage to locate salient features and thus automatically define common sense concepts. However, the efforts to date have failed to do so sufficiently to empower more intelligent systems. Thus, I suggest that we explicitly encode our current knowledge "by hand" so that our systems have it available and can overcome their current limitations.

[4]Eventually, modeling of behavioral processes will be realized on a neurological level; for the present, a step in the right direction is the construction of symbolic models. Any model of processes and structures is more worthwhile than circuitously trying to guess satisfactory properties for the musical surface, which is simply an artifact of the structures and processes.

[5]We are currently limited to those ideas which we can articulate, but the use of computer programs to describe the structures and processes of understanding is expanding our vocabulary.

[6] By recognition of music I mean the determination of musical structure, the detection of salient surface patterns and high-level qualitative descriptions, etc. By reproduction I mean the ability to produce music that is similar to a given music according to these recognized important elements.

[7]Generation tasks first require listening abilities (in order to play music, you must be able to hear it, in order to evaluate your output). Thus the current design focuses on listening, and includes a less developed description of the possibilities for generation.

[8]These networks can also produce some of the same plan-following behavior [Maes 90] as traditional centralized systems, and can be adapted to hierarchical organization structures suggested by studies in ethology [Tyrrell 93].

[9]The term "music theory" is generalized here to include other types of work (beyond academic music theory) which have theories of music, ranging from music cognition to jazz method books.

[10]Meyer uses a definition of emotion roughly as the existence of an affective state (contrasted, for instance, with "cool, calculated observation"); regardless of the debate about the quality of this definition, his discussion of emotion in music is useful and relevant, for a different definition can be substituted for his if need be.

[11]A similar evaluation of research relating possible specific brain mechanisms to qualitative musical properties such as affect [Clynes 82; Pribham 82] is of interest here, but is beyond the scope of the current work.

[12]These are similar to the operations I implemented in the DBX system, but more general. The DBX system had one way in which to realize each of these effects, while the current system has several; thus the present concepts are more richly represented.

[13]Trademark aspects of Armstrong's playing style such as his innovative use of the trumpet and timbre are not considered here; I consider only MIDI information such as pitches and rhythms here. (The inventiveness of his improvised tunes provides sufficient material for study.) Musical Instrument Digital Interface (MIDI) is a simplifying protocol for note information about a score, providing information similar to common practice notation. The signal processing issues which would be necessary if not working with MIDI are a separate research topic than the issues addressed here. Playing in MIDI streams in real-time allows for the simulation of real-time cognitive processing, and is necessary for producing output in a performance situation.

A MIDI stream is a representation of music which focuses upon the onset and release times of pitches and their absolute velocities. Although the actual MIDI specification includes facilities for handling a myriad of other data as well (continuous streams of numbers as "controller data", or larger user-defined segments of data as "system exclusive messages"), many users in the computer music community have been dissatisfied with its limitations, particularly with regard to the absence of a useful representation of timbre. Many interactive generative and analysis systems have still adopted MIDI because of its wide support and ease of use. In fact, the MIDI representation of music as a list of note onsets (with a velocity/loudness) is the primary representation in many systems; these systems usually add only segmentations of the list, and calculate the intervals between successive pitches.

[14]Questions of beat-tracking can be separated from the more general questions of musical knowledge focused upon in this project. Beat-tracking is a difficult research problem [Desain 92; Rosenthal 92] in its own right, and will not be addressed here. I provide my system with "the beat", and assume that agencies could be designed which would extract this information from the regularities of the rhythm section. Thus, this system works with rhythm data essentially like that indicated by common practice notation. The general representational framework, however, allows for future work on understanding timing deviations as performance expression. Also, I believe that the general approach towards musical intelligence presented here could be of use in considering the beat-tracking problem.

[15]These ideas can be found throughout SOM, in particular in the sections on Level-Bands and Polynemes. Additional specific concepts (such as the use of Paranomes to extend similar roles into several realms) appear in the system as a natural result of using shared functionality between classes in a network of agents.

[16]This use of the term "agent" is a combination of Minksy's and Maes' use of the word; from Minsky comes the idea of an architecture made from specialized agents, while from Maes comes the use of agent as module in a spreading activation network for action selection. This is distinct from the idea of a "software agent", or the other uses of the term.

[17]Within a sequence, the preceding and following notes to a current note can also serve as pre-/post-conditions in terms of spreading activation, as described in the *Processes of Playing* section.

[18]Inheritance and other features of C++ are used here to allow the general characteristics of base classes to be shared by many specific types of classes. These features are not described in this document; the reader unfamiliar with the basics of object-oriented programming is referred to any of the language reference manuals available on the subject.

[19]This use of the term "frame" is roughly based on Minsky's definition [Minsky 75], but is basically similar to a *class* structure in an object-oriented language; the important feature is that the concept has slots that are filled in by default values. To relate this to Minsky's terms: concepts are referred to in memory by *K-lines*, which are pointers to these frames. The saved values of the slots are K-lines to particular states of these agents; thus they are labels of the instance from the perspective-realm of the agency that computes the value. For example, an apple could be represented by a frame with slots for a set of K-lines, each of which activates agents of color, taste, shape, weight, etc. to be in a particular state. In music, a phrase is represented by its pointers to its consituent notes, pairs, groups, reductions, etc. My use differs from Minsky's in that I make several instances of frames in order to save the state values of agents; in addition, I do not use the more powerful concept of frame-arrays.

[20]The current set of knowledge can be expanded by using the system's representations as tools in music cognition experiments or as atoms for computational learning algorithms in order to discover higher level qualities and categories of music that can be quantified. The work presented in this thesis provides the framework for that endeavor.

[21]There are no extra-musical elements in this discussion of the elements of music, although most humans' understanding of music includes such elements. To include extra-musical features in a significant sense would require general progress in AI which has not yet occurred. While it is possible that this is a fatal flaw in the system, one which will prevent it from understanding fundamental aspects of music, it appears from the current investigation that purely intra-musical features provide plenty of information to achieve a much higher degree of musical intelligence and sophistication than is currently available in other computer music systems.

[22]In the DBX system, I reduced the number of parameters to require only one value for each of syncopation, metric level (activity), and cross rhythm. The result was a system which could produce rhythms roughly characteristic of the intended values indicated by each of the parameters. By further developing and extending this simplified set of parameters into the more complete representation in the current system, I enable all rhythm patterns to be in the range of the generator, and I make explicit the relationships between them on different skeletal levels (as discussed in the *Melodic Line Knowledge* section). By default, notes which are not aligned with an metrical downbeat are heard as anticipations of the downbeat rather than delayed occurrences of the downbeat; however, a strong specific cycle level can create an expectation which reverses this default assignment.

[23]Note that because the PitchSeries is defined on numbers from 0-11, actual pitches such as F (MIDI number 65) to G (MIDI number 67) in the key of F must first be shifted to align them with the default key of C (F - C = 5; 65 -> 60, 67 -> 62); then, they must be reduced to generic degrees (pitch mod 12: 60->0, 62->2); these values of 0 and 2 are then the values that may be in the list.

[24] The slot is usually used to hold a larger structure (through a pointer) than simply an integer value. Because it is only accessed by external functions through an internal function which has a generic external appearance, external functions can use it without concern for its specific definition within the class

[25] The current calculations to fill in these slots are trivial, because the system is using MIDI and polls a demon to find out the current harmonic/metric context. The quantization is also simple, placing the note in the nearest metrically defined slot; this is satisfactory for development, but would need to be amended in a performance system. The architecture can accommodate more complex agencies, such as for the detection of MIDI features from an acoustic signal as well as descriptions of articulation and timbre, if used with a module that provides such quantified descriptions [Egozy 95].

[26]This is analogous to, but more general than, the activity and syncopation parameters in the *textures* of the DBX system.

[27]Where there are gaps in between strong sequences, default filler material can be assumed. This material is simply the conventional metric hierarchy of emphasis combined with scale step or arpeggiation motion, as described in the *Background Knowledge* section.

[28]"Reductions" and "groups" are terms used by GTTM (and elsewhere in music theory) to describe the structures postulated above the level of the musical surface. While my approach has been influenced by the music theory literature regarding these concepts, my adaptation of these ideas is not strictly in accord with the convention meaning of the terms. This is a result of the function which my representations are serving: my representations both need to conform to musician's concepts about the music and to actually be quantifiable to a degree such that implementation is possible. Thus the more malleable definitions used in music theory are maintained here only in their motivational definition; the specifics are quite different.

[29]A useful intuitive example of a reduction is the coercion of the important notes of an eighth-note phrase onto a half-note phrase: the result has fewer pitches but probably covers the same general ground. Reduced lines seem

more appealing as representations of a phrase when they themselves follow smooth melodic paths -- although an uneven path may actually be a more accurate skeleton of the phrase, and is in fact what the system uses as its type of reduction.

[30]Note that the system is not interested in deciding a specific resolution for overlapping conflicts; the goal here is to represent all relevant evidence, and then have each interpretation be used according to its strength. See *Appendix 2* for further discussion on the fundamentally ambiguous nature of determining the relation of heads to elaborating events in simple rhythms.

[31]This distinction between reduced skeleton as intrinsic representation and distinctive features as intrinsic representation has been frequently neglected or confused in the discussion of music theory reductions applied to computer systems; my approach subsumes both options.

[32] This is in keeping with the notion of general understanding described above, from [Minsky 86]. In music, [Lerdahl 85] succinctly states that comprehension takes place when the perceiver is able to assign a preceise mental representation to the stimuli. Human listening involves a myriad of general orientation, categorization, and attention focusing skill before we get to the level of understanding disturbances of air as instances of music [Minsky 86]. The discussion here is a symbolic representation of the process of translating recognized notes into learned concepts, a process which humans might perform with quite different operations; the goal is to model the process sufficiently to be able to generatively predict its artifacts. The general thinking skills which are necessary for humans to perform this limited task in context are in the vast category of unsolved fundamental problems of artificial intelligence.

[33] This organization of sensors is currently implemented by *message broadcasting* between agents. This simply means that agents are linked by placing the higher level agent in a list of recipients of the lower level agent's messages; the link is a soft one, that can be changed during execution of the program by adding or removing agents to other agents' lists. A similar functionality could be achieved by spreading activation along these links, where a threshold of activation was required to enable the higher level agent to look for its occurrence.

[34]At each step, where the description says that new instances are created, the actual process first checks to see if an identical instance already exists; if so, that instance is used and is considered to be the "new" instance by increasing its Activation and creating pointers to it from the rest of the new representations. This prevents having the NoteFrame agent point to countless instances of the note D, for each phrase that uses a D can point to the same D instance. However, higher level frames are less frequently matched exactly, so new instances for these are usually created.

[35]Ambiguity between interpretations need not always be resolved, for conflicting representations can be maintained in the system. However, when there is a high threshold for the amount of Emphasis required to maintain a representation, there is effectively competition between instances. In these cases, ambiguity is "resolved" as a result of the continuing spreading of Emphasis, which eventually makes one instance dominate the others; this is similar to the mechanism which Minsky terms *cross-exclusion.*

[36]The traversal of a scale can be seen as a learned convention in that the scale fragments which constitute a whole scale are repeatedly found in old material, and thus the scale itself as an expected path arises out of the combined effect of the partial matches with old fragments. The current system has no mechanism for turning this generalized feature into a learned concept, and thus a scale path is only expected in terms of literal matches to saved instances or a continuation of a feature value.

[37]Phrases which appear to be completely new to a listener are usually variations upon known structures. The point is simply that a phrase need not be thought of as a spontaneous gesture which arises out of nowhere; rather, it can be explained in terms of the influences and constraints which lead to a set of possibilities that are chosen between based upon desired quality. When we say that Armstrong was freely inventing melodic ideas, we mean simply that his combination of the possible structures and constraints was not restricted to elaborations of the song's tune.

[38]The spreading activation occurs in a continuous loop; when a note is ouput, the loop is interupted for the listening processes to evaluate it, and then the generation processes resume.

[39]The intuitive appeal of this approach is that it reflects the fact that in humans, the features themselves originally have to be learned from sets of instances; here, the features are literally represented by the sets of instances that influence their occurrence.

[40]The implementation of this system has been difficult. As each section of code was developed and tested, it revealed an aspect of the design that needed to be reconsidered. This process is normal over the doctoral timeline for which this project was originally intended, but has led to compromises over the shorter master's timeline. The design has reached a stable point at present, where it reflects my approach and manages to achieve its functionality.

[41]The experiment's focus was the exploratation of how the "head" of the pair could be determined: the head is that element which is seen to create a period on a higher level, such that instead of noticing a repeating period on the level of the tones, the subjects would "tap" or otherwise indicate that the period between the first members of each pair or between the second members was salient. Their finding was that by varying the time interval between the two notes, the subjects' interpretation of which of the two was the head could be shifted. Hearing the first note as a weaker "upbeat" to the stronger second corresponds to the "progression" relationship of GTTM, whereas hearing the second as a weaker "afterbeat" to the stronger first corresponds to the "prolongation" relationship.