On the Use of Problem Reduction Search for Automated Music Composition

Stacy C. Marsella and Charles F. Schmidt
Laboratory for Computer Science Research and Department of Psychology, Busch Campus, Rutgers University

Abstract

The expertise required to recognize musical structure or generate acceptable musical compositions has proven to be very difficult to capture within a computational model. This research addresses this issue from an unusual perspective. Our work in this area has grown out of an interest in understanding the control issues that arise in the use of the method of problem reduction. Musical composition was selected as one domain in which to experiment with problems of control because of the highly complex structure apparent in musical composition. This chapter discusses a problem reduction approach to automated music composition. The application of this approach to the task of writing melodies is then discussed and some examples are shown.

In Balaban, M., Ebcioglu, K., and Lake, O. (Eds.) *Understanding Music with AI: Perspectives on Music Cognition*. Cambridge, MA: AAAI Press/MIT Press, 1992.

Chapter 10

On the Use of Problem Reduction Search for Automated Music Composition

Stacy C. Marsella and Charles F. Schmidt
Laboratory for Computer Science Research and Department of Psychology, Busch Campus, Rutgers University

A musical composition is a highly structured artifact. An expert in the musical genre of the composition can decompose it into parts and recognize at least some of the various types of relationships between the parts. The expertise required to recognize musical structure or generate acceptable musical compositions has proven to be very difficult to capture within a computational model. Our own research addresses this issue from an unusual perspective. Our work in this area has grown out of our interest in understanding the control issues that arise in the use of the artificial intelligence (AI) search method known as *problem reduction* [Nilsson 1971, Bresina et al. 1987]. Musical composition appealed to us as one domain in which to experiment with problems of control because of the highly complex structure apparent in some musical composition (e.g., [Lerdahl and Jackendoff. 1983, Tenney 1986]).

One of the most apparent features of musical compositions is the presence of hierarchical structure. This has led some researchers to explore the adequacy of formal grammars as a basis for capturing some of the structural features of musical compositions (e.g., [Laske 1972, Lidov and Gab. 1973, Lerdahl and Jackendoff. 1983, Roads 1985a]). Grammars are particularly attractive formalisms by which to attempt to capture the structural properties of musical compositions because they allow for the generation of an infinite set of strings from a finite vocabulary, and because their use in generation or parsing assigns a hierarchical structure to the resulting terminal string. The non-terminal nodes of this hierarchical structure or tree provide a formal basis for representing the fact that the music is organized into larger and more abstract groupings such as phrases, contours, themes and the like. It is quite clear, however, that a context free grammar is not powerful enough to

capture all the apparent structure that can be discerned in most musical compositions. In particular, the use of repetition of note, motive, or theme, either unaltered or under some transformation at discontinuous locations in the string, suggests a structure of dependency across the "tree of derivation," as well as the hierarchical dependency captured in the tree.

A dependency that holds between nodes of the tree of derivation that are not dominated by a common parent node will be referred to as a horizontal dependency. A concern for such horizontal dependencies is evidenced in much of the musical grammar research. So, for instance, Holtzman employed a formal grammar more powerful than context free, an unrestricted grammar [Holtzman 1980]. This concern is also evidenced in researchers employing representations influenced by AI, such as [Ebcioglu 1988].[1] As we shall see, such dependencies also pose interesting problems for the control of problem reduction search because the context is non-local, in effect, it extends "arbitrarily" between nodes in the tree of derivation.

A simple musical example of such a horizontal dependency can be seen in Beethoven's *Écossaise in G* shown in Figure 1. This piece can be analyzed as a two-part or binary form. In this AB structure, the first eight measures constitute the A section and the final 8 the B section. Note the similarity of the last two measures of the A section and the last two measures of the B section. Horizontal dependencies of this type cannot be naturally expressed within a context free grammatical formalism, and, indeed are difficult to generate efficiently within any AI search method, including the computational framework provided by the standard problem reduction search method. Our interest in extending the problem reduction search method to efficiently handle the generation of solutions that exhibit such horizontal dependencies motivated our use of music, more specifically melodic aspects of music, as a domain in which to study the control problems posed by such horizontal dependencies.

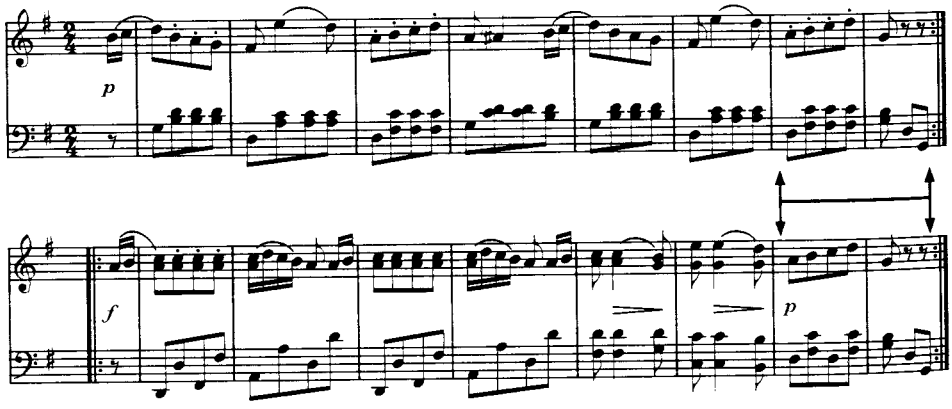

Figure 1. Excerpt from Beethoven's Écossaise in G

Problem Solving and Musical Structure

This computational approach led us to consider the way in which musical composition might be cast if it is viewed as the result of a problem solving process carried out within the framework provided by the method of problem reduction. In this view, the problem solver, or "composer," begins with two sets of specifications. The first is a partial specification, usually quite abstract, of the musical material that is to be used in forming a musical composition. Call this the *starting state* of the composition problem. The second is again a partial specification, also usually quite abstract, of the requirements that the musical composition should successfully satisfy. Call this the *goal state* of the composition problem. Using this pair, the problem solver makes some pattern of choices that transforms the starting material into a sequence of notes and their associated timing that satisfies the goal requirement. A history of this pattern of choices and the associated context within which these choices were made provides a structural representation of the composer/problem solver's decisions.

From this viewpoint, it is important to note that whatever structure is discernable in the sequence of notes is the result of, but may only partially reflect the structure of, the decisions made by the problem solver. Thus, the structure of the composition itself is viewed as homomorphic to the structure that is reflected in the history of the decisions made by the problem solver in forming the composition. It should be recognized that this implies that the decisions made by the problem solver cannot necessarily be recognized unambiguously from any analysis applied to the sequence of notes generated by the problem solving process. From this point of view, the study of musical composition and its various forms can be understood as the study of the control structure of the corresponding problem solving process rather than the study of the structure of the resulting musical sequence of notes. This point of view will not be defended. Rather it should be viewed as a general hypothesis about a useful way in which to understand musical structure. The work described in this chapter represents some preliminary research which begins to provide a basis for evaluating the plausibility and usefulness of this hypothesis.

With regard to this hypothesis, it should be noted that under a generic interpretation of "problem," there is considerable work that views composition as either a problem solving process or a problem of making decisions (or choices). For example, see [Gill 1963, Hiller and Isaac. 1963]. In contrast, two points should be stressed about the present work. First, the problem reduction model provides an explicit, and technical, definition of what constitutes a problem [Nilsson 1971]. Second, the intent, and consequences, of seeing composition as a problem solving process are different. This research is concerned with the control of problem reduction search. To this end, our concern is the *history* of control decisions made in that search, as opposed to just the solution tendered, or a hierarchical representation of that solution free of control issues. It is the maintenance of this history, or pattern, of control decisions, not just the choic-

es, that is critical in the control of the problem reduction search.

The Method of Problem Reduction

The specific problem solving method we use to explore musical structure is problem reduction. This method involves recursively decomposing a complex problem into presumably simpler subproblems until only primitive subproblems remain whose solution is "obvious". The complex problem is solved if the solution to each of the primitive subproblems jointly imply a solution to the original complex problem. This condition trivially holds if each of the subproblems is independent of all other subproblems. Subproblem independence rarely holds. Consequently, the focus of the research on this problem reduction method has been on ways to recognize and solve problems where there exists some dependency between subproblems. Some of the areas that have been studied using this search method include the so-called robot planning problems [Fikes et al. 1971, Sacerdoti 1977], various puzzle domains [Amarel 1981], and the domain of logical proofs [Amarel 1967].

Because the method of problem reduction involves searching for a solution by recursively decomposing a problem, the history of the decompositions that yield a solution can be thought of as assigning a structure to that solution. However, the interest in AI research on problem reduction has typically emphasized the way in which this model can be used to *control* the decisions that must be made in searching for a solution to a given problem, rather than on the *structure* that such a model "assigns" to the solution created by this model. This focus on control naturally arises because problem reduction is studied as a general method for conducting a heuristic search for a solution to some problem. Further, there may typically be several ways in which to derive a particular solution as well as many possible solutions to a given problem. Consequently, there is no bias that dictates that a given solution should have a unique structure assigned to it. Additionally, in the problem reduction model, rather than viewing the starting state as some generic symbol as in a context-free grammar, there is the idea of satisfying or deriving a given goal from some specification of starting assumptions or resources. Consequently, a given goal coupled with differing starting assumptions or constraints will typically yield differing solutions.

The problem reduction method can be abstractly described as involving a cycle that includes a *decomposition* phase and a *composition* phase. The decomposition phase involves the recursive application of a process of problem decomposition. The structural representation of this process defines a hierarchically organized AND/OR tree of subproblems where the decomposition of a problem into subproblems is represented as an AND node and alternative decompositions as OR nodes. As in the case of the top-level problem, each subproblem also consists of two types of specifications. This pair of specifications will be referred to as the starting state and the final or goal state of the subproblem.[2] Together this state

pair provides the context for the subsequent attempt to find a solution to the given subproblem. A decomposition rule, or in the parlance of problem reduction, a *non-terminal reduction rule*, rewrites the given problem state pair into a set of such problem state pairs. *Terminal reduction rules* serve to recognize primitive subproblems. A problem is solved when all terminal nodes of an AND tree in the AND/OR tree have been recognized as primitive, and the composition of each set of child subproblems implies the solution of their parent subproblem. Such an AND tree constitutes a *plan* to solve the problem.[3]

In order to explicate the control issues involved in problem reduction it will be useful to depict some of the ways in which control can be achieved in problem reduction search. To this end, a graphical depiction of a partially expanded AND tree is provided in Figure 2. Here the top or root node represents the start of the problem solving process. Associated with this node is the problem specification which is depicted as a black oval. The decomposition of the problem specification into two ANDed subproblems is represented by the nodes immediately dominated by the root node. The left-most subproblem is shown as further decomposed into two additional ANDed subproblems each of which is itself decomposed into two further subproblems. Each of these subproblems happens to be primitive and thus undergoes no further decomposition. The "wide" rectangles dominated by each of these terminal nodes represent a terminal partial solution. The composition of each of these pairs yields the partial SubSolution represented as "tall" rectangles associated with the parent node. These are then composed in turn to yield the SubSolution to their parent problem. Thus, at this point in the depiction of the search, the decomposition phase has been recursively applied to the leftmost portion of the tree followed by the recursive application of composition to yield the SubSolution to the left subproblem associated with the overall problem.

Note that the search has proceeded in one of the many possible orders in which the subproblems might be expanded. If all of the subproblems are independent then the order in which the frontier of this tree is expanded is irrelevant. However, subproblem independence is the exception rather than the rule. Consequently, the focus of AI research in problem reduction search has been on ways of detecting and efficiently handling subproblem dependence. Most of the approaches to handling subproblem dependencies have involved a control structure where subproblem independence is assumed to hold during the decomposition phase of this cycle, and then the validity of this assumption is tested during the composition phase [Fikes et al. 1971, Sacerdoti 1977]. The models of problem reduction search vary in the type of mechanism provided to deal with subproblem dependencies that are discovered during this composition phase. Our own approach to the development and implementation of a model of problem reduction search has involved extending the expressive power of the rules of decomposition to allow the nature of the dependency between subproblems to be specified. This information is then used to control the problem solving process. It is this ability to express the specific nature of subproblem dependencies and the usage of this knowledge in the control of the problem solver that we will exploit in our experiments in the domain of music composition.

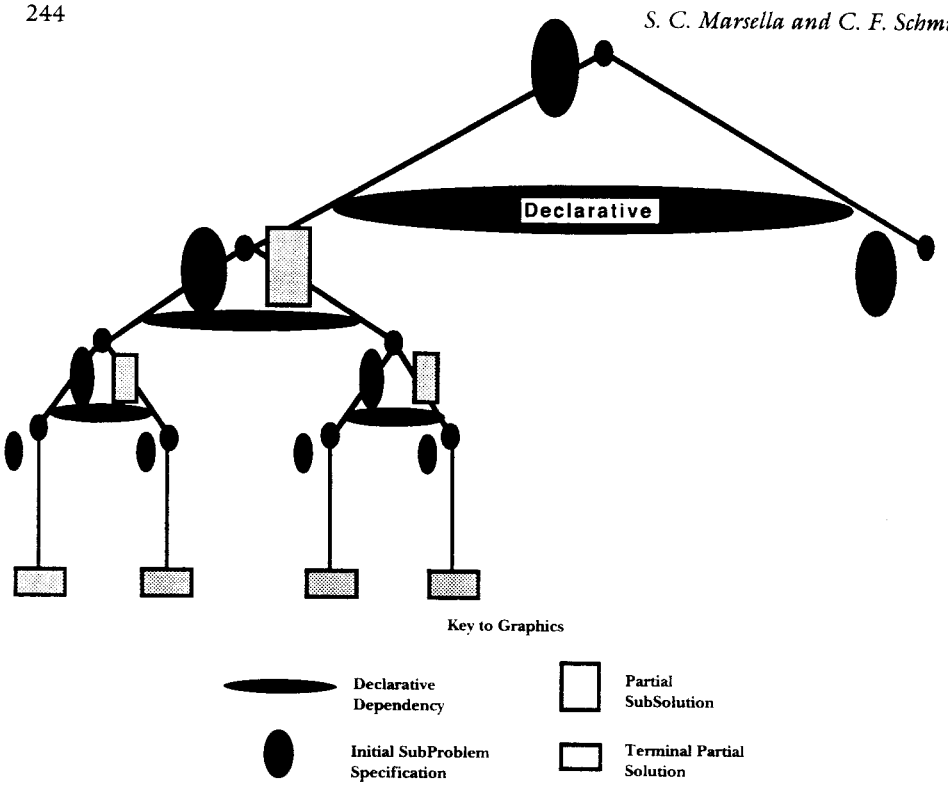

Figure 2. A Partially Expanded Problem Reduction AND Tree with Declarative Dependency

By specifying the contents of subproblem state pairs, a rule can indirectly affect the way each of its descendant subproblems will ultimately be solved. By indirectly affecting how the subproblems are solved, this specification can be employed to predictively handle dependencies between those subproblems. Furthermore, a subproblem high in the hierarchy of decompositions can leave a trace of its contributions to the surface structure of those terminal elements of the solution that it dominates.

The predictive specification of the contents of descendent subproblems is graphically depicted in Figure 2. Note the "wide" black ovals linking the sibling subproblems. This depicts the fact that the subproblem specification allows for the *implicit* expression of dependencies between the descendent subproblems. These are labelled a *declarative dependency* in the figure because their expression is based on how a given rule rewrites a parent problem into the subproblem state pairs (see [Marsella 1988] for further discussion). However, as we shall see, not all subproblem dependencies can be handled predictively within this hierarchical structure.

This explicates the relation between hierarchical structure present in the solution and the hierarchical history of problem decomposition. However, it does not explain how a terminal element of the tree, or more generally, a partial subtree, can affect parts that are not dominated by that subtree. This is the situation that holds in the description of the excerpt from Beethoven's *Écossaise in G* discussed

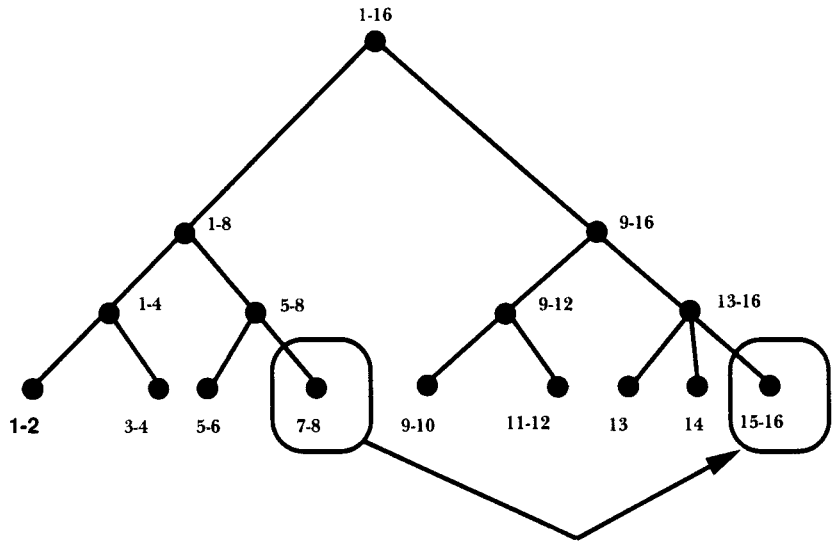

Figure 3. A Structural Representation of an Excerpt from Beethoven's Écossaise in G.

earlier and shown in Figure 1. A graphical representation of a simple structuring of this excerpt is shown in Figure 3. The numbers associated with each node refer to the measures dominated by that node. The node labelled 1–8 represents the A part, that labelled 9–16 the B part, of this binary form. Here the horizontal arrow points out the dependency between the node labelled 7–8 and that labelled 15–16. Note that node 7–8 is dominated by the node that represents the A part and node 15–16 by the node representing the B part. The only node that dominates both these nodes is the root node.

Dependencies of this type can be realized within a problem reduction method if an ordering is imposed on the expansion and solution of the subproblems in this tree. The imposition of an ordering on the solution to the subproblems allows aspects of the solved subproblems to be passed up the tree and across to disjoint branches of this tree. In this way the results of the solution to one subproblem can affect the way in which a quite distant subproblem is solved. In terms of the current example, this result can be achieved if it is known, at the time that the root node is expanded, that the terminal element of the solution to the "A" subproblem of the binary form is to be used as material for the terminal solution of the "B" subproblem. This can be achieved if the problem solver first completely solves the "A" subproblem, passes *aspects* of this solution to the "B" subproblem, and then essentially solves the "B" subproblem in a more or less "right to left" fashion so that the solution to node 15–16 can appropriately influence the solution to the remaining nodes in the "B" subproblem.

The important point is not the specific ordering suggested, but rather that a specific ordering is required. To arrive at the musical structure of this example

using a control strategy which does not in some sense "know" the ordering requirements at the time that the top node is expanded would be extremely difficult.

One limited approach, that extends the ability to control problem reduction search, exploits the ability to order the solution of subproblems. This is graphically depicted in Figure 4. In this case, the subsolutions are passed back up the tree during the composition phase. These partial subsolutions are depicted as "tall" rectangles. These, then, in turn can provide additional specification of the remaining subproblems. The "wide" gray rectangles represent the *solution derived dependencies* that can be resolved once the subsolutions have been resolved. These *solution derived dependencies* can be employed by virtue of the ordering that has been imposed on subproblem expansion. In this figure, the ordering is from left to right so that leftmost subproblems are expanded first. As denoted by the curved gray lines, the entire solution for a subproblem composed during the composition phase is passed up the tree. This subsolution is used to specify the dependency between that subproblem and a subproblem in an adjoining branch of the tree. This dependency is then employed to constrain the search for a solution to that as yet unexpanded subproblem.

There is an inherent limitation in the use of solutions to specify dependencies that is of particular relevance to musical composition. The solution must be *interpreted* in order to derive the dependency. For instance, reconsider the graphical representation of the excerpt from Beethoven's *Écossaise in G* shown in Figure 1. As analyzed, the dependency is between the node labelled 7–8 and that labelled 15–16. The dependency is not between the entire solution to the A part (the node labelled 1–8) and the B part (the node labelled 9–16). To express the actual dependency between nodes 7–8 and 15–16 minimally requires details of the structure imposed on the two parts (nodes 1–8 and 9–16) during their generation. This information is not present in the solutions to those parts. It would have to be recovered, or derived by analysis of the solutions.[4]

This limitation suggests an alternative approach that further extends our ability to control problem reduction search. Consider Figure 5. Here again, an order is imposed on expansion. However, instead of passing a solution constructed during the composition phase, a structure termed a *design artifact* is built during the plan generation or decomposition phase. The design artifact is passed between subproblems (now termed a Design SubArtifact Specification) to express a *plan derived dependency*. In the figure, these are depicted as "wide" gray ovals that span sibling subproblems. As we shall see in the section titled "Experiments," this design artifact includes information about the decisions made in the generation of a subsolution that can be readily employed to specify dependencies between that subsolution and discontinuous subproblems.

The problem reduction method that we have developed, REAPPR (Reformulation And Parallel Problem Reduction), is designed to allow the use of these various mechanisms for specifying dependencies in controlling problem reduction search [Bresina et al. 1987, Marsella 1988]. Rather than discuss REAPPR abstractly, we will now turn to a discussion of its application in writing simple melodies.

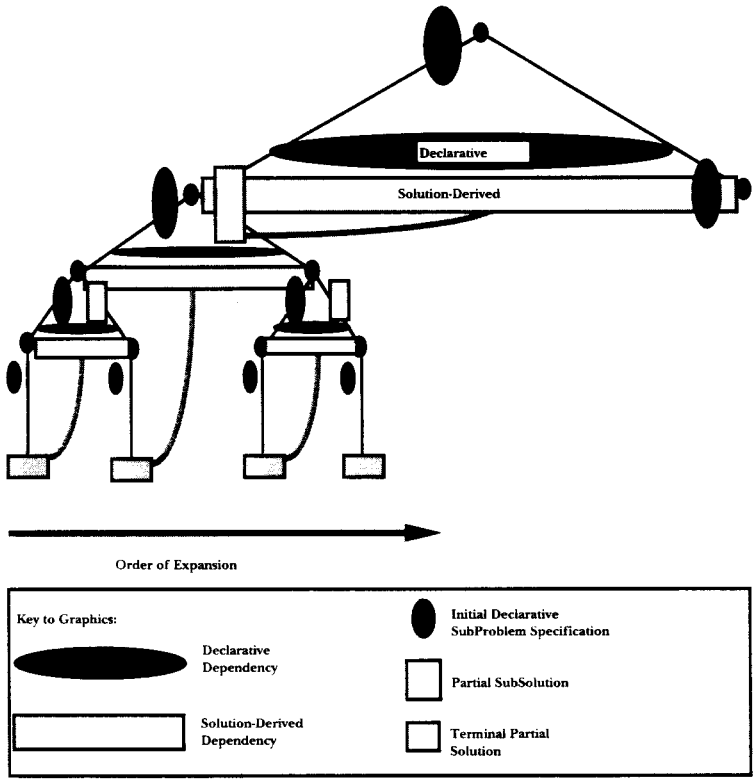

Figure 4. A Partially Expanded Problem Reduction AND Tree with Solution Derived Dependency Expressed through SubSolution Passing

Experiments

A series of simple experiments are being conducted as an initial illustration of our problem reduction approach to melody composition. The task for these experiments is to generate simple melodies using a fixed set of forms (e.g., rounded chanson, rondeau, etc.). In addition, stylistic constraints are placed on the rhythm, melodic motion, and tonal organization (e.g., tritone leaps were to be avoided, large leaps or chromatic intervals were to be used only rarely, major phrases should end on the tonic or dominant, etc.).

The problem solving process used in all the experiments proceeds by decomposing the composition task into a COMPOSE subproblem and a CRITIQUE subproblem. The major task for COMPOSE actually involves building and manipulating the *design artifact.* The design artifact represents those aspects of the composition structure that must be communicated in the plan (i.e., the AND tree) in order to handle horizontal dependencies. Thus, it represents a relation between parts of the composition

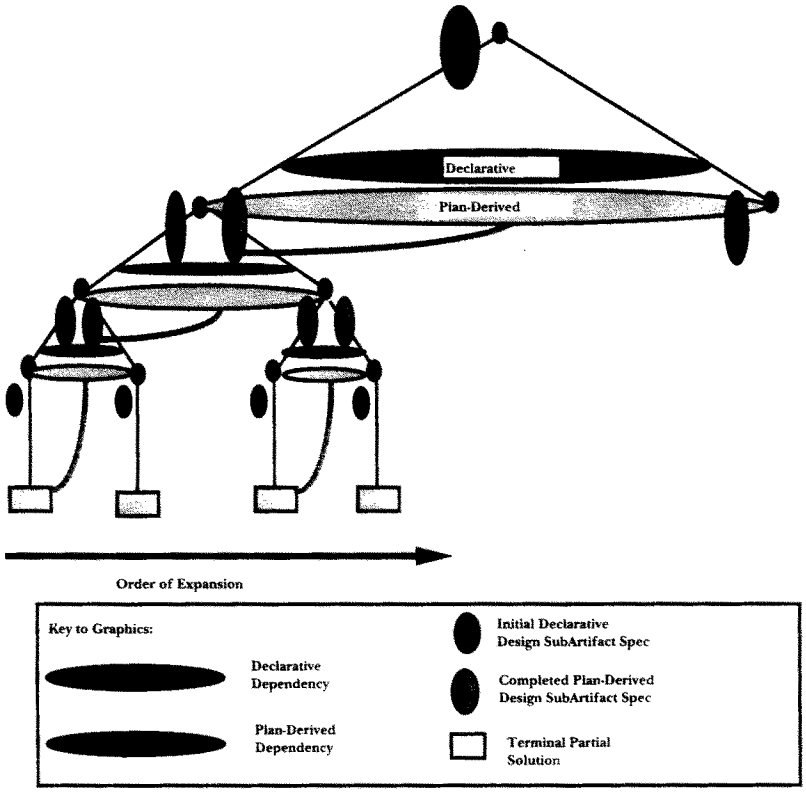

Figure 5. A Partially Expanded Problem Reduction AND Tree with Plan Derived Dependency Expressed through SubArtifact Passing

structure and the control decisions involved in creating it that permits a more predictive handling of the horizontal dependencies.

Originally these experiments used an interval based approach to melody writing. One of the first "melodies" (entitled here as "First Melody") composed by REAPPR under this approach is shown in Figure 6. Figure 7 shows some of the overlying structure that dominates parts of that melody. A left to right ordering of the frontier of the complete structure would recreate the melody shown in Figure 6. Note how the piece is broken into an **AABA'** structure at the highest level. In these experiments, structure forming rules that typically one would envision only being applicable at the highest level of the structure could also be recursively applied at lower levels. In keeping with this possibility, Figure 7 illustrates how the **A** is broken into **A1**, **A1**, **A2**, and **A1'** parts.

Building this structure also involves several subtasks. Tonal centers (as well as portions of the overall scale) are assigned to these phrases and subphrases. This assignment is based on the phrases' role in the overall melody, their depth in the hierarchy, and their relation to their "parent" phrase. So, for instance, the large

intervallic leaps seen in the **B2** part are related to the fact that it is the middle of its parent section's **B1 B2 B3** structure which in turn is a middle section in the overall **AABA'** form.

The repetition and variation dependencies seen in this example are quite simple. Indeed, the lateral dependency suggested by the repetition of sibling parts was actually generated by REAPPR by virtue of its ability to specify the order (and repetition) of solution subsequences. It did not require non-hierarchical communication. Furthermore, the variations implied by the **A'** and **A1'** parts did not fully materialize in this piece due to rule selection decisions made lower in the hierarchy.

In more recent experiments, we have been moving to harmony based approaches to melody writing. These new experiments are better exemplars of the problem reduction control issues we are exploring, in part because the structure of the design artifact requires a more elaborate dependency structure in the plan that generates it.

A partial representation of a strongly dependent artifact is shown in Figure 8. In this partial design artifact, the overall melody is composed of three "sub"melodies (typically eight bar periods). The first of these submelodies is labelled Part A. In turn, Part A is composed of two submelodies (typically an antecedent and consequent, which has an implied musical dependency between them) labelled **a** and **b**. As opposed to such hierarchical structure, there is a lateral dependency between Part A and its sibling, Part Ar. Part Ar is a repetition of Part A. A more interesting lateral dependency is seen in the case of Part B, which is a "variation" built from one of Part A's submelodies, **b**. Thus Part B must be composed by modifying the sub-structure that describes **b**. This lateral dependency cuts across the hierarchy and between levels of that hierarchy. Thus, fragments of melodies (e.g., motives) can be employed at different points in the hierarchy as a source for variations and repetitions.

Besides fixing a pattern of repetition and variation, these lateral dependencies suggest the necessary control and flow of information for the problem reduction process. That is to say, the rules, and, in particular, a plan generated by those rules, must in some effective fashion deal with these dependencies. For instance, Part A must be planned prior to Part B and Part Ar. Moreover parts of the design artifact created for Part A must be provided to composition tasks which generate the artifact structure for Part B.

In order to understand the role this artifact would play in communicating dependency information in the plan tree, we will look at a rule that builds and employs the artifact. Figure 9 presents a stylized representation of part of such a non-terminal rule. At the top of the rule is the problem specification which describes the conditions under which the rule can be applied. This rule decomposes the problem of creating a melody based on an **ABA** pattern into three distinct subproblems. These subproblems require different kinds of knowledge in order to be solved and are in fact solved in different reduction spaces. In essence, this is a knowledge intensive approach that is modeling composition in terms of the design decisions that must be made. As denoted by the lateral arrows, *plan derived dependency* information

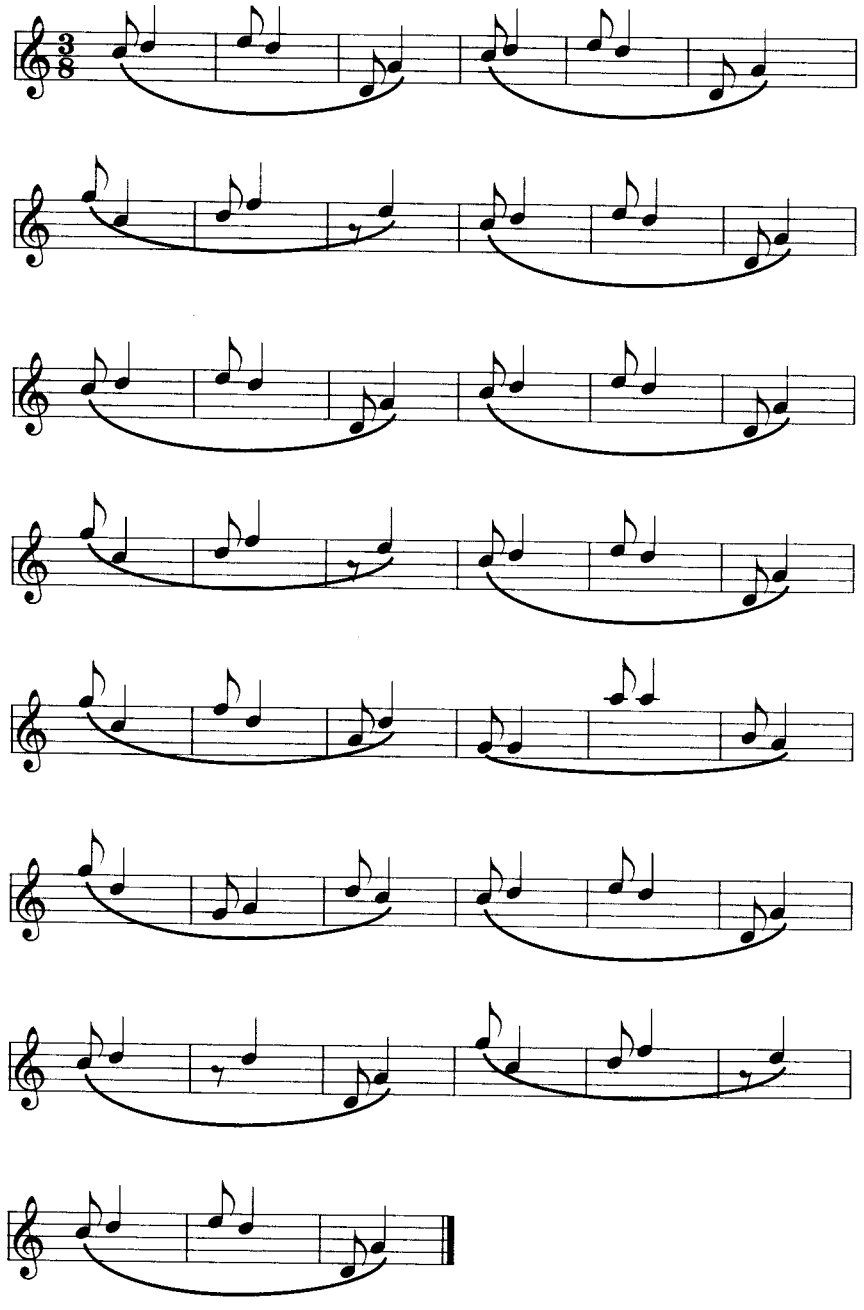

Figure 6. REAPPR's "First Melody"

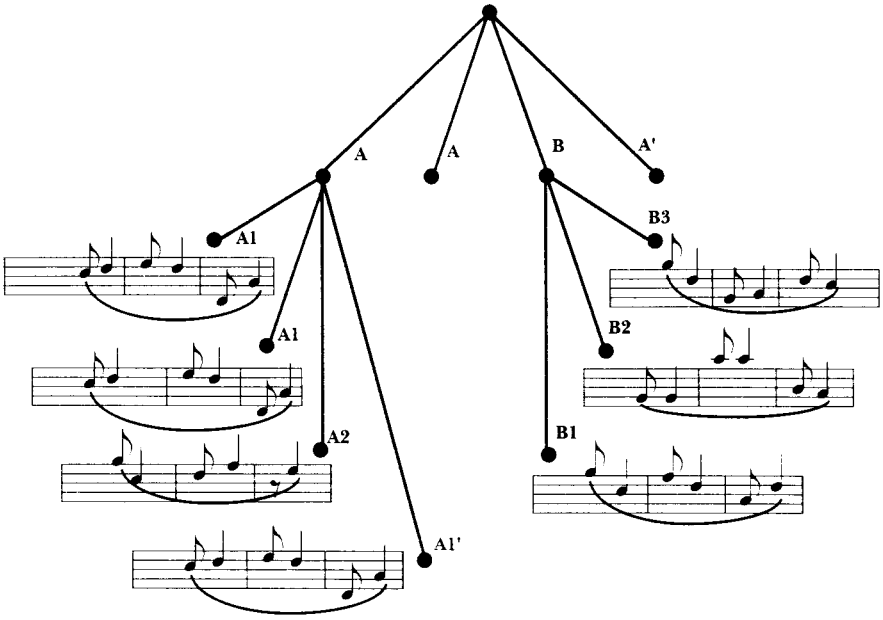

Figure 7. Partial Representation of REAPPR's "First Melody"

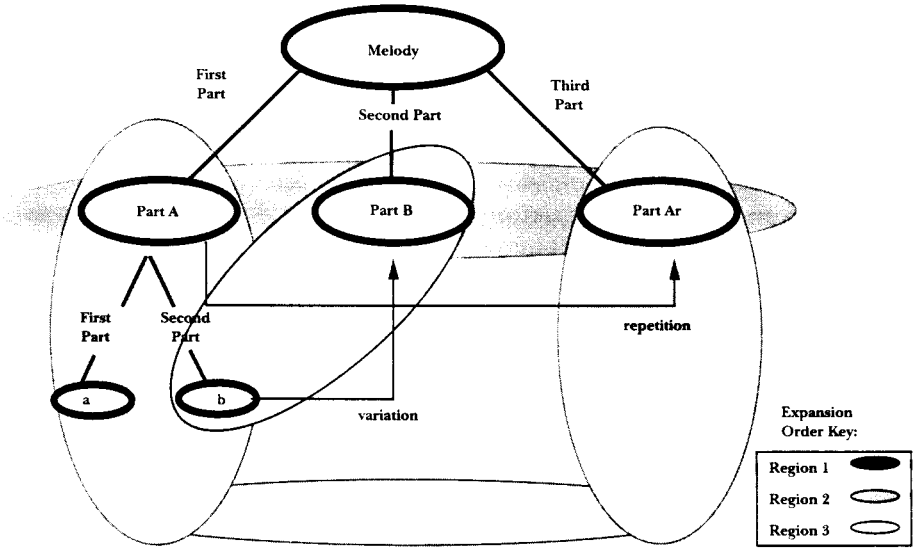

Figure 8. A Few Details of a Partial Design Artifact

acquired during the generation of a solution to one subproblem can be employed to further specify a subsequent subproblem. In other words, the order in which subproblems are solved is part of the design decisions represented in the rule and determines how the various subproblems constrain each other. For instance, note the partial orders specified at the bottom of Figure 9 and the relation to the specifications of the subproblems. Subproblem1, which derives the chords and melodic contour, precedes the COMPOSE subproblems and as such constrains the solutions of those subproblems.

The structure of the decomposition and the segment of the plan it would create are related to, but distinct from, the artifact depicted earlier in Figure 8. Although not explicitly depicted in Figure 9, the design artifact of Figure 8 is specified incrementally, as the search for solutions to the subproblems proceeds. And, this evolving design artifact in turn provides the local context required for the solution to the remaining open subproblems. The goal state of each subproblem determines what aspects of the design artifact it will reference as well as how it will modify/enhance the artifact. Relating the two figures, the DERIVE subproblems of Figure 9 modify the artifact by more fully specifying MELODY. In Figure 8, this is depicted as Region 1. Both COMPOSE subproblems build a hierarchical grouping structure. COMPOSE **A** takes the result of the DERIVE subproblem and builds additional structure that describes **A**. In the artifact, this additional structure corresponds to Region 2. COMPOSE **B** builds a variation by viewing just the **b** part built by the COMPOSE **A**, as depicted by Region 3 in Figure 8.

Note that this relation between **A** and **B** achieves several ends. The **B** subproblem's view of the artifact is restricted to just part of the overall state of the artifact. This localization is an important approach to generating subproblems whose solutions are simpler to find [Lansky 1986, Marsella 1988]. Furthermore, the dependency between the subproblems is predictively specified, or handled, at the time the rule is applied. In order to do this, **B** used knowledge of what **A**'s structure would be. This is not always available and it can be more difficult to specify predictively when these variations or associational structures[5] occur across widely disparate levels of the artifact. In such cases, either backtracking in the planning process or opportunistic planning may be required.

The DERIVE subproblem must be decomposed by additional rules which capture musical constraints, such as resolution to the tonic, melodic motion, etc. In addition, at the level of each phrase (**A** and **B**), the selection of rhythmic patterns is performed. In part, these derivation decisions are based on the role a melodic phrase plays in the overall melody, the depth in the design artifact at which that phrase is positioned, and its relation to "parent" phrases. However, at this juncture of the experiment, the knowledge brought to bear on these derivations is quite primitive.

Additional rules would grow (i.e., recursively extend) the plan (and consequently the design artifact) until it eventually terminates at a frontier which tentatively represents the notes of the melody. The Critique subproblem evaluates/modifies the artifact based on stylistic constraints (e.g., avoidance of tritones) and the goals implicit in the hierarchical view of the structure.

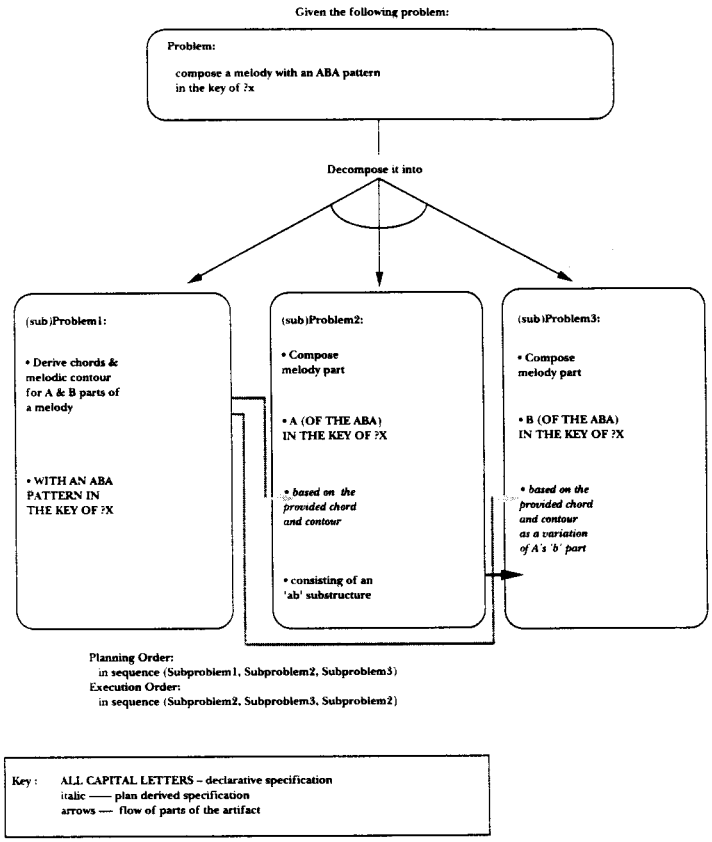

Figure 9. Partial Representation of a Rule for an ABA Pattern

Discussion of the Experiment

The preliminary computational experiment encourages our hypothesis that compositional expertise can be captured within this model of problem reduction. And, this model appears to provide a principled way in which to explore how various patterns of control create the surface structure of a musical composition.

Two structures, the problem reduction AND tree and the design artifact, lie at the heart of how this research explores the relation between patterns of control and surface structure. The full AND/OR search tree is created by recursive application of reduction rules that can express a variety of control patterns via the orders imposed on the search for subproblem solutions and the corresponding state specification mechanisms. This expressive power allows the decisions made in forming a musical composition to be modeled, and if need be, to be modeled at a fine grain. So, for instance, depending on how a rule is written, a melodic contour could be fixed for

the entire melody or just a part of it. The contour can also be described abstractly and refined/modified later. And, different rules can potentially place different orders on when the contour decision is made relative to other decisions.

While the AND tree provides a hierarchical history of the recursive application of rules, it is the design artifact which must be introduced to allow for the efficient handling of horizontal dependencies. The design artifact provides the structure for selectively grouping information from various parts of the current state of the AND tree and then communicating this information to open subproblems in the tree. As depicted in Figures 8 and 9, this allows the composition process to be modeled effectively with problem reduction searches which take place in specialized search spaces. Each of these searches can access and build upon a different part or view of the artifact. The ideal is that the reduction search is provided with a local view of the relevant decisions that have been made. At the same time, information and constraints are communicated between the reduction searches by the effect each search has on the artifact. Finally, there is an overall strategy for controlling and ordering the subspace searches as well as for amalgamating and refining the partial results of those searches.

The design artifact thus provides more than a hierarchical description of the eventual surface structure. It also allows for encoding the parts of the design decision history which impose constraints on the solutions to the design subproblems. Different views of the design artifact selectively communicate these constraints across subproblems solved in different spaces. This allows the rich sets of constraints and interactions that a subproblem solution builds to be more readily incorporated by another subproblem's view.

The declarative specification of the design artifact provides a basis for comprehending what the system is doing. This comprehension is useful for both the system and its user. It allows the system to model decisions, and gives it the ability to modify and replan musical structures. For a human user, it opens the possibility of interacting with the system. In so far as the goals expressed in the design artifact are realized in the resulting melody, the artifact also provides a basis for evaluation.

Concluding Remarks

AI approaches to problem solving and design tasks typically involve some form of search through a space of alternative decisions or choices. Effective realization of such search paradigms typically depends on controlling the decisions which are taken, or considered (or re-considered in the case of systems that backtrack). So for instance, in exploring the use of AI search techniques in music production, researchers have explored the use of intelligent or dependency directed backtracking and search control heuristics (e.g. [Ebcioglu 1988, Ames 1987]).

The use of a problem reduction search paradigm in this research provides several

benefits for the design task. By reducing a problem into a collection of design subproblems, the reduction approach allows a global view of the search for a completed design even though the search for solutions to those subproblems proceeds locally. Aspects of the overall design, and the role the subproblem solutions will play in that design, can be embedded in local searches, thus controlling them and reducing the need for backtracking. This ability is exploited when building the rich set of relations that provides for the assignment of tonal centers and scales, as well as the patterns of repetition and variation. This global view is in contrast to other search techniques such as state space search which tend to be constrained to a very local view of the ongoing search.

An interesting comparison can be made with the CHORAL expert system that harmonizes and performs a Schenkerian analysis of chorales in the style of J.S. Bach [Ebcioglu 1988]. The system employs a rich representational structure, views of which are tied to, and serve to structure, a rule based generate and test search. Based on these views, the search follows a repeated fixed control cycle of chord skeleton, fill-in, Schenker-bass, and Schenker-descant.

In contrast, our interest in studying control issues that arise in the use of problem reduction led us to explore the elevation of aspects of the search control to the level of the expertise encoded in the problem reduction rules. Thus control becomes an explicit, flexible part of the search decisions/compositional process. This interest in control issues is reflected in the fact that it is not so much musical knowledge that ends up being modeled, but rather knowledge about how to control the compositional process (cf. [Laske 1989]). A major concern for this research is whether problem solving expertise as captured in this problem reduction paradigm will elucidate structural issues in musical composition. At the present time, the approach looks promising, but final results must await further experimentation.

Acknowledgments

The authors gratefully acknowledge the help of John Bresina in the early phases of this research.
This work was supported in part by the National Science Foundation under Grant No. DCR83-18075.

Notes

1) See [Roads 1985b, Ames 1987] for surveys on the application of AI techniques to music.

2) When problem reduction is applied to the domain of planning sequences of physical actions [Nilsson 1971], the starting state and goal state each constitute a partial description of the state of the world. As we apply problem reduction to this music domain, the states typically refer to more abstract aspects of the musical design rather than the surface elements of the resulting composition.

3) The reader should be aware that some of the AI planning literature uses the term *plan* to denote just the solution. The distinction between the two uses of the term should become more salient as the discussion progresses.

4) This limitation is not peculiar to this domain or task. For instance, even in simple "Blocks World" robotic domains [Nilsson 1971, Sacerdoti 1977], planning often requires that the solution, a

sequence of robot actions, has to be re-interpreted as changes to the state of world.

5) Or, in the more general case, when any dependency occurs across widely disparate levels of the structure.

References

[Amarel 1981] Amarel, S. 1981. Problems of representation in heuristic problem solving: Related issues in development of expert systems.In *Methods of Heuristics*, eds. Groner, Groner and Bischof. Hillsdale, N.J.:Lawrence Erlbaum.

[Amarel 1967] Amarel, S. 1967. An approach to heuristic problem solving and theorem proving in the propositional calculus. In *Systems and Computer Science*, eds. J.F. Hart and S. Takasu. University of Toronto Press.

[Ames 1987] Ames, C. 1987. AI in Music. In *Encyclopedia of Artificial Intelligence*, ed. S. Shapiro, 638–642. John Wiley & Sons.

[Bresina et al. 1987] Bresina, J.L., Marsella, S.C. and Schmidt, C.F. 1987. Predicting subproblem interactions, Technical Report LCSR-TR-92. Laboratory for Computer Science Research, Rutgers University.

[Ebcioglu 1988] Ebcioglu, K. 1988. An Expert System for Harmonizing Four-Part Chorales. *Computer Music Journal*, 12(3):43–51.

[Fikes et al. 1971] Fikes, R.E., Hart, P.E. and Nilsson, N.J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.

[Gill 1963] Gill, S. 1963. A technique for the composition of music in a computer. *The Computer Journal*, 6(2):129–133.

[Hiller and Isaac. 1963] Hiller, L. and Isaacson, L. 1963. Experimental Music. *The Modeling of Mind; Computers and Intelligence*, K. Sayre and F. Crosson, 43–71. Notre Dame, Indiana: University of Notre Dame Press.

[Holtzman 1980] Holtzman, S.R. 1980. A Generative Grammar Definition Language for Music. *Interface*, 9:1–47.

[Lansky 1986] Lansky, A.L. 1986. A representation of parallel activity based on events, structure, and causality. *Reasoning about Actions and Plans, Proceedings of the 1986 Workshop*, eds. M.P. Georgeff and A.L. Lansky. Los Altos: Morgan Kaufmann.

[Laske 1972] Laske, O. 1972. On Musical Strategies with a View to a Generative Theory of Music. *Interface*, 1:111–125.

[Laske 1989] Laske, O. 1989. Composition Theory: An Enrichment of Music Theory. *Interface*, 18:45–59.

[Lerdahl and Jackenfoff 1983] Lerdahl, F. and Jackendoff, R. 1983. *A Generative Theory of Tonal Music*. Cambridge, Mass.:MIT Press.

[Lidov and Gab. 1973] Lidov, D. and Gabura, J. 1973. A Melody Writing Algorithm Using a Formal Language Model. *Computer Studies in the Humanities and Verbal Behavior*, 4(3/4):138–148.

[Marsella 1988] Marsella, S.C. 1988. An Approach to Problem Reduction Learning, Technical Report LCSR-TR-110, Laboratory for Computer Science Research, Rutgers University.

[Nilsson 1971] Nilsson, N.J. 1971. *Problem Solving Methods in Artificial Intelligence*, 80–115. New York: McGraw-Hill.

[Roads 1985a] Roads, C. 1985. Grammars as Representations of Music. *Foundations of Computer Music*, eds. C. Roads and J. Strawn, 403–442. Cambridge, Mass.: MIT Press.

[Roads 1985b] Roads, C. 1985. Research in Music and Artificial Intelligence. *Computing Surveys*, 17(2):163–190.

[Sacerdoti 1977] Sacerdoti, E.D. 1977. *A Structure for Plans and Behavior*. New York: Elsevier.

[Tenney 1986] Tenney, J. 1986. *Meta-Hodos and META Meta-Hodos*. Oakland CA: Frog Peak Music.