An intelligent tutoring system for tonal counterpoint: From process to structure

Panayotis Mavromatis
Department of Music and Performing Arts, New York University, USA
Affiliated Faculty, Computer Science Department, NYU Courant Institute, USA
panos [dot] mavromatis [at] nyu.edu - http://theory.smusic.nyu.edu/pm

Matthew Brown
Music Theory Department, Eastman School of Music, University of Rochester, USA
http://theory.esm.rochester.edu/matthew_brown/

Proceedings of the fourth Conference on Interdisciplinary Musicology (CIM08)
Thessaloniki, Greece, 3-6 July 2008, http://web.auth.gr/cim08/

Background in music theory. Although classical musicians are expected to master the principles of tonal counterpoint, they find them notoriously difficult to learn. In part, these difficulties arise because traditional counterpoint tutors have two shortcomings. First, most tutors give students elaborate lists of rules, but they do not give advice on how students should organize those rules in specific contexts. Second, traditional tutors tend to formulate the rules in terms of prohibitions: students often learn what not to do than what to do. As a result, they fail to develop effective procedures for creating good counterpoint.

Aims. This paper addresses these issues in the framework of Intelligent Tutoring Systems. An Intelligent Tutoring System (ITS) is an interactive computer environment that teaches students how to solve problems in a specific domain. Building an effective counterpoint ITS requires techniques from music theory, Artificial Intelligence, and cognitive psychology. This synergy can produce pedagogical tools of unprecedented power and flexibility. But it can do more: the system can also yield new insights into the nature of human expertise and its acquisition.

Background in Cognitive Psychology and Artificial Intelligence. Recent psychological research suggests that experts outperform novices because experts 1) focus less on the individual elements of a problem and more on large patterns and 2) develop strategies for resolving conflicts between different constraints, for navigating around obstacles, and for backtracking if necessary. Experts display their prowess by developing rich vocabularies to describe their methods. Distilled from hundreds of hours of practice and used consciously, these vocabularies are normally classified as *declarative knowledge*. Experts also devise specific strategies for solving problems. Hard to describe verbally, rarely addressed in explicit instruction, and requiring many hours to develop, these strategies fall under the rubric of *procedural knowledge*.

Artificial Intelligence models expert problem solving through the concept of a *problem space*. A problem space encompasses all possible problem configurations, or *problem states*, required to reach a solution. Problem states are transformed into other states through *operators*; these operators define the possible actions taken to solve a problem. Modeling begins by enumerating the objects and relations explicit in the problem description. Expert problem solvers expand this basic problem space to create an *augmented problem space* that includes their store of declarative and procedural knowledge.

Main Contribution. The ITS has three components: an expert system that implements the various rules of counterpoint; a graphical user interface that allows the student to interact with the system; and a system monitor that tracks the student's individual problem-solving steps and offers feedback along the way. Within this ITS framework, implementation of expert problem solving is carried out in two stages. Stage 1 develops a procedural model for teaching tonal counterpoint, which we call 'The *Lodovico* Method.' The model begins by structuring the concepts and rules that explain the behavior of tonal voice leading and harmony. It then derives a list of concrete steps that students can follow in order to harmonize a simple tonal melody. These steps invoke a finite list of transformations that allow them to proceed from one problem state to another. Stage 2 builds an augmented problem space, whose states are defined through the concepts and rules of Stage 1. In this problem space, operators correspond to the transformations identified in Stage 1. This implementation allows one to 1) evaluate the effectiveness of 'The *Lodovico* Method' by computer simulation of problem solving as search in the augmented problem space, and 2) subsequently implement 'The *Lodovico* Method' in the form of ITS-generated feedback to the student.

Implications. The research presented in this paper highlights the usefulness of an ITS as a tool for formalizing and evaluating music theories. In addition, an ITS can be used to collect detailed data on how experts and novices perform a problem-solving task. As such, an ITS allows researchers to build and fine-tune models of musical expertise. Finally, the ITS approach has implications for understanding musical structure: by showing how experts structure their knowledge of musical constraints, and how they internalize this knowledge in effective procedural strategies, the present analysis suggests that musical structures should be understood as products of expert behavior rather than as static systems of constraints.

Although classical musicians are expected to master the principles of tonal counterpoint, they find them notoriously difficult to learn. In part, this is because the process always requires a lot of time and effort; there are unfortunately no shortcuts to success. But the situation is made even worse by the fact that traditional counterpoint tutors are not always up the task. They usually fall short on two counts.

First, most tutors give students elaborate lists of rules, but they do not give advice on how students should organize those rules in specific contexts. For instance, students may be unclear about when to let a line move by step (local rule) and when to deviate from that rule in order to shape the line as a whole (global rule). We refer to this conflict as "The Rule Structuring Problem".

Second, traditional tutors tend to formulate the rules in terms of prohibitions: students often learn what not to do rather than what to do. Students often become frustrated as they try to solve problems by a process of unguided trial and error; they waste considerable time performing and subsequently undoing unnecessary actions. To help students achieve effective results as quickly and efficiently as possible, the teacher needs to systematize the procedural aspects of creating good counterpoint. This is what we call "The Procedural Problem".

This paper proposes to address these issues in the framework of Intelligent Tutoring Systems. An Intelligent Tutoring System (ITS) is an interactive computer environment that teaches students how to solve problems in a specific domain (Forbus & Feltovich 2001; VanLehn 1999; Polson & Richardson 1988). In a typical ITS session, the system presents students with a given problem and then monitors their progress towards a solution. The system must be able to understand what students know and must respond to questions and requests for help accordingly; it must be able to determine when to offer unsolicited advice; and it must be able to exercise control over the curriculum by selecting and sequencing material to suit the students' needs.

Building an effective counterpoint ITS requires sophisticated input not only from music theory, but also from Artificial Intelligence (AI) and cognitive psychology. Music theory provides the concepts and rules that define a given task, as well as explicit procedures for completing it effectively. Meanwhile, AI provides the design principles needed to build complex computer systems that are able to represent and manipulate knowledge in various domains (Nilsson 1998). Finally, cognitive psychology provides insights into human knowledge, reasoning, problem solving, learning, and skill acquisition (Anderson 1993, 2000; Ericsson 1996; Newell & Simon 1972; Singley & Anderson 1989).

If successful, the payoff of this project is enormous. One of the obvious outcomes will be a powerful and flexible tool to meet various pedagogical demands. Perhaps equally importantly, the project will offer unique insights into the nature of human expertise and its acquisition. The ITS can, in fact, assume a central role in this research, offering an experimental platform for collecting data and testing theories of music structure and cognition.

Modeling expert problem solving

What constitutes expert skill in counterpoint writing? Even though knowledge of the counterpoint rules is clearly necessary, there is strong evidence to suggest that it is not sufficient. On the one hand, the rule statements for each counterpoint task typically occupy a few pages at most and can be memorized in a short period of time. On the other, the acquisition of expertise in counterpoint is a long and painstaking process, often involving several years of practice.

The problem space formalism

The general theory of expertise (Ericsson 1996) suggests that the difference between experts and novices seems to be that experts 1) recognize and respond to big patterns of problem material rather than individual elements and 2) develop strategies for resolving conflicts between different constraints, for navigating around obstacles, and for backtracking if necessary.

Experts display their knowledge in two different ways. First, when asked to describe how they solve problems, experts have rich vocabularies in which they characterize the patterns and strategies that they use. These vocabularies are distilled from hundreds of hours of practice solving problems and not just from explicit instruction. Since such knowledge can be consciously accessed and described, it is classified as *declarative knowledge* by cognitive psychologists (Anderson, 2000). Second, some aspects of expert knowledge are manifest in patterns of problem-solving behavior. Hard to describe verbally, rarely addressed in explicit instruction, and requiring many hours to develop, this knowledge is usually referred to as *procedural knowledge* (Anderson, 2000).

The preceding remarks provide a methodology for analyzing expert problem solving that characterizes expert knowledge structures and makes them available to explicit instruction. Originally developed by Newell and Simon (1972), this methodology still features prominently in the development of tutoring systems (Anderson 1993).

Fundamental to Newell and Simon's approach is the concept of *problem space*. The problem space encompasses all possible problem configurations, or *problem states*, required to reach a solution. Problem states are transformed into other states through *operators*; these operators define the possible actions taken to solve a problem. Modeling begins by laying out the *basic problem space*; this roughly corresponds to the objects and relations explicit in the problem description, such as musical notes and the intervals between them. However, expert problem solvers expand the basic problem space to create an *augmented problem space* that includes the store of special knowledge that they possess.

Building a problem space

There are two modeling techniques that can be used to build an augmented problem space. The first technique involves studying expert problem solving in real time, an approach pioneered by Newell and Simon. In this approach, experimental subjects are presented with a problem to solve, and the experimenter records the step-by-step process of problem solving. In addition, subjects are requested to "think aloud" throughout the process, describing the rationale for each of their actions, a technique known as *concurrent verbal protocol*. Analysis of the solution steps reveals the structure of the subjects' problem space, in terms of 1) what operators are available to them and 2) the context in which they prefer to deploy each of these operators. In addition, analysis of the verbal protocol in conjunction with the solution steps reveals the concepts and strategies characterizing the subjects' augmented problem space, which in this way formalizes and models the subject's level of expertise. In the case of music, it might be possible to determine these concepts and strategies by detailed studies of composers sketches and revisions (Brown 2003).

The second modeling technique, which forms the main focus of this study, is to formalize the music theorists' intuitions in a theoretical procedural model that simulates some idealized problem solving behavior. In a sense, the theoretical model constitutes a music theorist's systematization of their pedagogical goals, which is often only implicitly conveyed in traditional instruction. This paper will focus on this second technique; it will outline 'The *Lodovico* Method' as a theoretical model for writing good counterpoint. The first modeling technique will also be briefly touched upon, but will be pursued more extensively in a future work.

Evaluating a problem-solving strategy

Once the augmented problem space is built using one of the above methods, or a combination of them, it can be tested in two different ways. The first way is to perform computer simulations that evaluate the adequacy of the problem-solving strategies embodied in that problem space. It is assumed that an effective strategy will navigate towards a solution using a minimum of backtracking and hopefully eliminate any haphazard aspect of the search towards a solution. By contrast, an ineffective knowledge representation will also lead to a solution, but only through "meandering" and

frequent undoing and redoing of its own actions, like a beginner who possesses no strategy and proceeds by random trials. This appraisal of a problem-solving strategy will be necessary in order to evaluate its instructional effectiveness and its independence from any additional procedural knowledge that the human expert may have that has not been formalized in the model. Results of such simulations will be reported in this paper.

A second way to test an augmented problem space is by studying its pedagogical effectiveness. Once a candidate problem-solving strategy has successfully passed the first test above, it can be implemented in an ITS that is not only cognizant of the successful strategy, but can also coach a student to use it. This means that, each time the student reaches an impasse and requests some help, the ITS must not only choose the correct operator to get the student out of the impasse, but must also explain to the student that choice. The explanation must be convincing and helpful, and must moreover lead to long-term improvement in students' problem-solving skills. Given a rationally constructed problem-solving strategy, explanation generation can be performed algorithmically by studying operator dependencies (Mellish 1990). Evaluation of the strategy's pedagogical effectiveness can then happen using standard experimental monitoring of the tutoring process. This second level of evaluation is clearly the most complex, and is beyond the scope of the present paper.

The basic ITS framework

The counterpoint ITS is written in Prolog; this language supports a declarative programming style where domain knowledge is directly represented in the form of logical relations and rules. The system resides on a central server machine. Individual student sessions take place through a graphical user interface (GUI), written in Java, that resides on a client machine and communicates with the server via the Internet. In its present form, the ITS allows the student to take one step at a time and gives feedback after each step.

Basic problem space

In this basic form, the ITS is able to recognize the structure of basic problem space. A state in this problem space is more or less characterized by the objects that appear on the score at a given stage in the solution. The basic objects are notes and rests. Each object is characterized by a set of attributes, like pitch and duration.

Operators in the basic problem space represent the elementary actions that can alter the problem state. In the basic problem space, three types of operators are defined, namely operators for *adding*, *deleting*, and *modifying* notes.

Following standard problem space formalism, operators are defined through preconditions and actions. Preconditions examine the problem state to determine whether a given operator is applicable. For instance, to add a note at a given location, the location cannot be already assigned a note. Actions implement the effects of the operator on the state. For instance, a "delete note" operator causes a given note to disappear.

Implementation of the counterpoint rules happens at the level of operator preconditions. For instance, in the case of an "add note" operator, the pitch and duration of the candidate note are examined to see if they satisfy the rules. If not, the operator is deemed inapplicable in the given state.

Operator selection

Determining the operators' applicability and choosing among a set of applicable operators is known as *operator selection*.

In *tutoring mode*, operator selection is performed by the student. At each state, they see the problem state represented graphically as a score. The operators corresponding to that state are presented as possible actions represented by GUI elements. The student must determine an operator's applicability. (Does the added note satisfy the rules of counterpoint?) If the student chooses an operator that's not applicable, the Tutor reports a violation (the most important counterpoint rules that are broken), and prompts the student to try

again. If the operator is applicable, the Tutor accepts it, modifies the problem state, and prompts the student to continue.

In *simulation mode*, the simulation determines which operators are applicable at each step, and if more than one is applicable, chooses one of them. If many different operators are equally applicable, the simulation chooses one at random. There are also ways to specify preferences among operators. For instance, it may be more fruitful to use an "add" operator more frequently than a "delete" operator. Such preferences are search heuristics. They add a realistic dimension to the search, and typically make it considerably more efficient.

This simple level of modeling clearly demonstrates the shortcomings of the basic problem space. In tutoring mode, it simply provides the student with an error check, but is unable to offer them strategic advice. In simulation mode, it is horribly inefficient, and arrives at a solution after a random walk through problem space. Clearly, a better structuring of the problem solving is needed.

Stage 1: `The *Lodovico* Method'

Although very much a work in progress, 'The *Lodovico* Method' helps to negotiate the two problems mentioned earlier. It addresses "The Rule Structuring Problem" by formulating a set of declarative rules of tonal voice leading and harmony that can be written in Prolog; it deals with "The Procedural Problem" by devising a sequence of steps that allows students to produce music that sounds tonal. To illustrate the basic ins and outs of 'The *Lodovico* Method', let us see how it helps students solve a typical problem in tonal counterpoint: harmonize a simple tonal melody.

'The *Lodovico* Method' makes certain basic assumptions about tonal voice leading. To begin with, it presupposes that tonal melodies move locally by step and globally by rising to a climax before descending back to the tonic for the final cadence. When two or more lines are presented simultaneously, they never move by parallel perfect octaves or fifths: instead, they move obliquely over a pedal tone, in contrary motion, or in strings of parallel thirds or sixths. The bass line can support these upper voices either by a simple pedal, stepwise lines in contrary motion, or by sequential progressions. The upper voices and the bass line are stacked in such a way that they only project complete or incomplete root and first inversion major, minor or diminished triads. These triads appear in functional strings, with dominant harmonies moving to tonics, and predominant harmonies moving to dominants. Figure 1 systematizes these observations by presenting a structured set of rules. These rules can be grouped into three domains (rules of melodic motion, rules of relative motion, and rules of vertical stacking) and classified into two main types (local rules and global rules).

Melodic Motion
Local
Lines mainly move by step.
Global
Lines converge onto ^1 at cadences.
Relative Motion
Local
Simultaneous lines do not contain parallel octaves and fifths between successive chord tones.
Global
Simultaneous lines contain strings of parallel thirds and sixths or contrary-motion patterns 12-10-8-7/6-5/T-3-U.
Vertical Stacking
Local
Tonal textures are built from complete and incomplete root and first inversion major, minor, and diminished triads.
Global
Tonal triads appear in functional progressions: dominants move to tonics and predominants to dominants.

Figure 1. The basic set of rules on which `The *Lodovico* Method' is built.

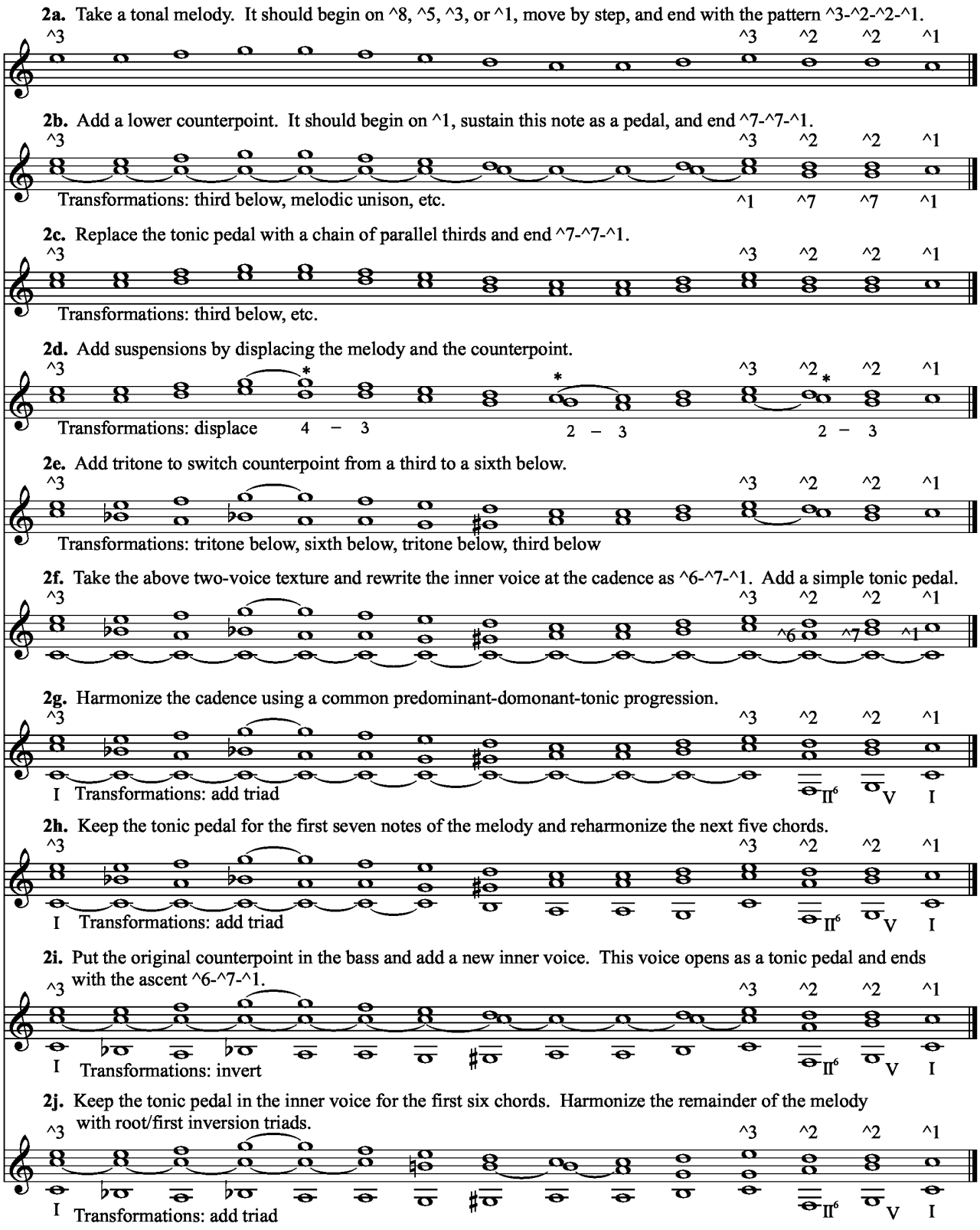

Figure 2. Simple melody harmonization using `The *Lodovico* Method'.

Figure 2 implements 'The *Lodovico* Method'. Figure 2a presents a typical tonal melody. It has a very simple overall structure: it starts in the tonic on ^3 and ends with a stepwise descent ^3-^2-^1 at the cadence. Locally, the melody strictly follows the rules of melodic motion, it moves exclusively by step. Next, Figure 2b adds a lower counterpoint to our original tune. This line consists of a long tonic pedal that extends to the cadence, where it converges onto the tonic ^7-^1. Figure 2c then composes out the tonic pedal with a chain of parallel thirds. Figure 2d spices up the chain of thirds with some suspensions. These are created by displacing one line against the other: the tune is displaced against the counterpoint at the first asterisk to create a 4-3 suspension, whereas the counterpoint is displaced against the tune to create a 2-3 suspensions at the second and third asterisks. The counterpoint can also follow a third or a sixth below the tune. To guarantee that this counterpoint proceeds entirely by step and remains perfectly invertible at the octave, Figure 2e includes a tritone to switch back and forth between thirds and sixths. In a similar vein, one could use *Lodovico* transformations to add a third voice, and gradually transform it through additional transformations, to reach an interesting three-part counterpoint. Figures 2f-2j show how this can be done.

For convenience, Figure 3 catalogs the various transformations used in the preceding examples. Following the layout of Figure 1, it arranges them into three types: those that control melodic motion, those that influence the relative motion of different lines, and those that coordinate the intervals between contrapuntal lines. In the first case, these transformations allow students to repeat a note (unison), add notes above or below another note, or displace one line over another. In the second, case, this chart includes transformations that invert or reverse contrapuntal lines. And in the third case, we have included transformations that allow us to harmonize a given melody. It should be stressed, however, that these transformations are constrained in a significant way: they can never produce parallel perfect octaves or fifths.

Melodic Motion
melodic_unison
step_up
step_down
third_down
fill
displace
Relative Motion
invert
reverse
Vertical Stacking
unison
third_below
sixth_below
octave_below
add_triad

Figure 3. Some transformations employed by `The *Lodovico* Method'.

Stage 2: Implementing *Lodovico* as augmented problem space

The augmented problem space was introduced earlier as a structure that models expert problem solving. Its two defining ingredients were identified to be the *concepts* and *procedures* that encapsulate expert declarative and procedural knowledge. 'The *Lodovico* Method' provides both of these ingredients.

An example of the former is the concept of a cadence. A cadence can be identified as pattern of notes that performs a certain function in the melodic line. In our framework, concepts are straightforwardly implemented as either new types of objects, or as additional properties of existing objects, and can be directly translated into Prolog code. Implementation of expert procedures is more sophisticated and will be addressed next.

Hierarchical subgoal decomposition

Studies of human problem solving show that experts master a complex goal by breaking it down into more manageable *subgoals*, each of which focuses on a specific aspect of the

problem. For instance, possible subgoals identified in *Lodovico* could be the writing of a cadence, the harmonization of melodic patterns with parallel sixths, etc. Each subgoal could in turn be broken down into smaller subgoals, which are thus organized into a *subgoal hierarchy*.

In our framework, subgoals are represented by *substates*. Substates are created by special operators, called *substate generators*. Such an operator's *preconditions* identify the conditions under which the subgoal is appropriate. If so, the operator selection process determines whether the subgoal will actually be pursued. The operator's *actions*, will then create the corresponding substate, which will put aside the long-term goal, and temporarily focus on the current subgoal.

Once a substate is active, the behavior specific to that substate can be defined through operators specific to that substate. Through the actions of these operators, the subgoal will eventually be attained, at which point the substate is automatically destroyed.

A *state stack* keeps track of the real-time dynamics of the problem-solving process's hierarchical structure. Each time a substate is created, it is pushed into the stack, and the higher-level state that launched it is temporarily suspended. When the substate is destroyed, it is popped out of the stack, and work in the parent state resumes.

Subgoal hierarchy in *Lodovico*

From the earlier discussion of *Lodovico*, it should be now clear how that method defines subgoals and their associated operators. In fact, the subgoals in *Lodovico* correspond to specific stages in the process, as represented, for example, by each individual staff in Figure 2. It is characteristic that each such stage has a simple high-level description, such as "Replace the pedal by a chain of thirds," or "Add suspensions." This high-level description becomes the basis of the subgoal's implementation in Prolog, with the help of the high-level concepts defined in *Lodovico* ("cadence", "chain of thirds", "switch").

The *Lodovico* operators available in the substates are precisely the transformations listed in Figure 3. These operators are defined as coordinated groups of basic actions, such as adding or modifying notes. Once in a given substate, focus on the task at hand can be maintained through substate-specific operator preconditions that only enable those operators appropriate to the given substate. For instance, within the "add suspensions" substate, all operators except "displace" are temporarily deemed inapplicable.

This implementation allows `The *Lodovico* Method' to be carried out computationally as problem-space search. In particular the method's effectiveness can be evaluated by computer simulation of problem solving.

Problem-solving simulations

Simulation A. Problem-solving simulations in basic problem space show that the latter requires thousands of moves in order to reach a solution. In this scenario, it feels like the simulation has reached a solution by accident, just because it could dedicate to it unlimited time and computational resources. Clearly our students do not possess either of these, and are therefore unlikely to finish the exercise, unless they are equipped with some strategy.

Of course, one could argue that this brute force search does not reflect even the lowest level of human performance. Indeed, preliminary analysis of experiments that record how people write counterpoint suggests that the vast majority of subjects have preference for two simple selection heuristics for operators of basic problem space, no matter how inexperienced these subjects may be.

First, people prefer to add new notes rather than modify or delete existing ones. This preference reflects people's basic optimism, since modifying or deleting is only necessary when a past choice leads to a dead end. For similar reasons, people prefer to modify rather than to delete notes; the latter option implies that no easy fix can remedy the impasse, and so drastic rewriting is required.

Second, people like to add new notes near positions that they have filled already, usually near the beginning or end of the given

melody. This is natural and desirable, since people tend to perceive a melodic line as continuous, and not as a patchwork of individual notes that can be filled in random order.

Simulation B. A basic problem space search that incorporates the above two heuristics now arrives at a solution within tens of moves, or a few hundred at most. The length of the search has been reduced by an order of magnitude. Nevertheless, the process still seems haphazard, and most of the steps taken appear unnecessary from an expert's point of view.

Simulation C. A problem space augmented with *Lodovico* heuristics shows a clear advantage over Simulation B. Depending on the length of the melody, problem solving now only takes 20-30 moves. Note that all *Lodovico* operators are compatible with the heuristics of Simulation B, since they mostly add notes, occasionally modify existing ones, and in fact never require any deleting. In addition, there is a clear preference for working in contiguous segments. However, *Lodovico* is more guided than Simulation B, because the above two heuristic principles are seen as a by-product of a rational plan.

Implications

Some pedagogical considerations

In evaluating the pedagogical effectiveness of theoretical models, one should of course keep one thing in mind: teaching students to simply follow instructions may not always be pedagogically effective. In fact, it is generally acknowledged that students learn best from their own failings. Therefore, in actual instruction, it should be expected that students spend a fair amount of time trying out things that don't work.

Nevertheless, the authors strongly believe that a procedural model like *Lodovico*, whose problem-solving efficiency is empirically tested, should always be around for the students to consult when their clumsy attempts lead them nowhere. In fact, it is precisely these failed attempts that will provide the motivational component essential to learning: after some wasted time and frustration, students will be more eager to try out 'The *Lodovico* Method', find out for themselves how effective it is, and hopefully internalize some aspects of it.

Of course, no student will ever behave exactly in accordance with a theoretical model. This is because such a model can only provide an idealized view of an expert's state of knowledge. Indeed, at any given moment, a student may have partially assimilated some theoretical principles, like *Lodovico*, while still using other heuristics of their own. The modularity of the problem space formulation in terms of independently available operators ensures that the formalism is able to represent the cognitive complexities of the actual learning situation as it evolves in real time. This is the principle of *student modeling*, one of the essential ingredients of the ITS approach.

Impact on the music theory curriculum

The effectiveness of one-to-one instruction has long been recognized by educational research (Bloom 1984). This is perhaps particularly true in music, where teaching individually or in small groups has been the dominant model of instruction for centuries. In a modern school, conservatory, or university setting, it may not always be practical to offer students the depth of individualized attention they may need. In such cases, an ITS can be a perfect complement to group instruction, monitoring and fostering each individual's growth. ITSs can also alleviate the work of the human teacher, taking on the burden of drilling basic skills outside the classroom to the extent needed by each student. Classroom time will then be reserved for the deeper and more creative aspects of instruction that only the human teacher can offer. In addition, by making their materials available on the machine, instructors will enable students to practice particular tasks as often as they want, at any time and from any point on the globe.

Understanding musical expertise

The ITS framework can form the basis of empirical studies in human learning. In the course of a tutoring session, an ITS can

gather data on how students go about solving problems by recording the order and timing of their actions. This feature offers researchers a unique view of the learning process, allowing them to evaluate the adequacy of procedural models such as `The *Lodovico* Method'. In this way, one can develop and refine theories of musical skill acquisition that will lead in turn to more powerful tutoring systems.

The task of writing counterpoint is a particularly apt choice as a study of expert problem solving. This is because the task is sufficiently narrow to allow the possibility of computer implementation, and yet it is rich enough to raise interesting compositional and aesthetic issues. In fact, counterpoint exercises have been an essential part of the composer's training since at least the 1700s; as is well documented, many major classical composers like Mozart, Beethoven, Brahms, and Schoenberg studied counterpoint and later used it in their teaching (Mann 1987; Salzer and Schachter 1969). This highlights the significance of counterpoint and suggests it is an ideal framework for studying the early development of compositional expertise.

Finally, this approach has implications for understanding musical structure. This is because the ITS shows how experts structure their knowledge of musical constraints, and how they internalize this knowledge in effective procedural strategies. For example, the extended *Lodovico* framework, which is not fully covered in this paper, highlights certain interconnections between sequences, pedals, and invertible counterpoint at the octave. This not only sheds light on the nature and procedural origin of certain commonly employed musical materials, but it also suggests that musical structures should be understood as products of expert behavior rather than as static systems of constraints.

Acknowledgments. The authors would like to thank Ken Perlin of NYU Courant for many stimulating discussions on pedagogy and human-computer interaction; Chris Brown of U of R, Department of Computer Science for his expertise in AI; Dave Millman of UNC Chapel Hill for expert work on GUI design. This work was supported by research grants from NYU Steinhardt and the University of Rochester.

References

Anderson, J. R. (1993). "Problem Solving and Learning." *American Psychologist* 48 (1), 35–44.

Anderson, J. R. (2000). *Cognitive Psychology and its Implications* (5th ed.). New York: Worth.

Bloom, B. S. (1984). "The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring". *Educational Researcher* 13, 3–16.

Brown, M. (2003). *Debussy's Ibéria.* Studies in Musical Structure and Genesis. Oxford: Oxford University Press.

Ericsson, K. A. (Ed.) (1996). *The Road to Excellence: The Acquisition of Expert Performance in the Arts and Sciences, Sports, and Games.* Mahwah: Erlbaum.

Forbus, K. D. and P. J. Feltovich (Eds.) (2001). *Smart Machines in Education: The Coming Revolution in Educational Technology.* Cambridge: MIT Press.

Mann, A. (1987). *Theory and Practice: Great Composers as Teachers and Students.* N.Y.: Norton, 1987.

Mellish, C. (1990) "Generating Natural Language Explanations from Plans." In Sterling, L. S. (Ed.), *The Practice of Prolog.* Cambridge, MA: MIT Press.

Newell, A. and H. A. Simon (1972). *Human Problem Solving.* Englewood Cliffs: Prentice-Hall.

Nilsson, N. J. (1998). *Artificial Intelligence: A New Synthesis.* San Francisco: Morgan Kaufmann.

Polson, M. C. and J. J. Richardson (Eds.) (1988). *Foundations of Intelligent Tutoring Systems.* Hillsdale, NJ: Erlbaum.

Salzer, F. and C. Schachter (1969). Counterpoint in Composition. New York: McGraw-Hill.

Singley, M. K. and J. R. Anderson (1989). *The Transfer of Cognitive Skill.* Cognitive Science Series. Cambridge, MA: Harvard University Press.

VanLehn, K. (1999). "AI and Education". In R. A. Wilson and F. C. Keil (Eds.), *The MIT Encyclopedia of the Cognitive Sciences.* Cambridge, MA: MIT Press.