Peter Kugel
Computer Science Department
Boston College
Boston, Massachusetts 02167 USA
Kugel@BCVMS.Bitnet

Myhill's Thesis: There's More than Computing in Musical Thinking

Introduction

Cognitive musicology (Laske 1988) is the study of musical thinking from a computational point of view. Like cognitive science, of which it is a part, cognitive musicology tends to focus on processes of musical thinking, rather than on products. Thus, for example, in the study of Beethoven's piano sonatas, cognitive musicology would not focus on the sonatas themselves, but on the processes that Beethoven used to compose them, that Serkin uses to perform them, or that a typical listener uses to listen to them. Like cognitive science generally, it tries to characterize these processes in computational terms.

Suppose, however, that there were aspects of musical thinking that could not be characterized in strictly computational terms. Would that mean that cognitive musicologists were wasting their time? The brief answer to this question is yes and no. The longer answer, which I want to develop in this article, involves a few more details and several digressions. I want to suggest that these details and digressions may be worth pursuing because, as I will argue in this paper, some musical thinking does require more than computations.

If this is so, then it suggests at least two things about musical thinking and our attitude toward it. First, it will not do to limit our concept of musical thinking to computations alone. If we want to fully characterize musical thinking in precise terms, we will have to use conceptual tools that are more powerful than computations. Second, musical thinking is—like scientific thinking by Popper's (1955) account—more open-ended and flexible than many people seem to believe.

Computer Music Journal, Vol. 14, No. 3, Fall 1990,

Myhill's Thesis

The claim that musical thinking cannot be wholly accounted for in computational terms seems to have been first proposed almost forty years ago by the late John Myhill (1952). Unlike people who claim that musical thinking cannot be characterized precisely at all—presumably because it involves "artistry" and "creativity," which elude scientific characterization—Myhill believed that musical thinking could be characterized with scientific precision. Myhill's proposal (henceforth referred to as *Myhill's thesis*) makes a positive claim to the effect that all musical thinking can be characterized scientifically or, as Myhill put it, with "crystal clarity." But it also makes a negative claim to the effect that certain aspects of musical thinking cannot be precisely characterized in terms of computations alone.

The negative part of Myhill's suggestion has been largely ignored for two reasons. One is that we seem to be amassing further evidence that it is false. As musical thinking is increasingly characterized in computational terms—see Ames (1987a; 1987b) for a review of how this is happening with regard to composition—it is becoming more difficult to see why anything should stop us from eventually characterizing it all in such terms. We might think of that as an empirical argument.

Another argument, which is more logical in nature, states that musical thinking is a kind of information processing; that it can be done by a physical object, the human brain. Since it is well known that any kind of information processing performed by a physical object can be simulated, or imitated, by a computing machine, it follows that musical thinking can be thus simulated.

Neither of these arguments is particularly convincing, and both are based on the same confusion—a confusion between practical and theoretical impossibility. Myhill's thesis claims that something is

theoretically impossible, but not that it is impossible from a practical point of view. In that respect, it is like the claim that the length of the diagonal of a square with sides one unit long cannot be precisely characterized by a rational number. The fact that—for all practical purposes—it can be, and that people can come up with more and more accurate rational approximations of its length, only shows that that length ($\sqrt{2}$units) can be approximated by a rational number for practical—but not theoretical—purposes. In much the same way, the fact that people, using computers, can approximate more and more of human musical thinking by means of computations has little, if any, bearing on whether or not such thinking can be fully characterized in terms of computations for theoretical purposes.

The argument that computing machines can simulate all information processing that machines can do well enough to "fool" people is no better founded than a parallel claim that $\sqrt{2}$ must be a rational number because we can produce a rational approximation so close to its real value that it would fool anyone.

Myhill's thesis does not bear on what we can do from a practical point of view. It bears on the conceptual tools we bring to bear on our attempts to characterize musical thinking, much as the mathematician's claim that $\sqrt{2}$ is irrational says little to the practical-minded engineer who continues to approximate it in terms of a rational approximation like 1.41. Just as the (theoretical) claim that $\sqrt{2}$ is not a rational number suggested to the Greeks that they should add such irrational numbers to their conceptual toolkit, so Myhill's thesis suggests that we should add "uncomputable processes" to the conceptual toolkit of the cognitive musicologist.

Myhill was rather vague about what such uncomputable processes might look like, but subsequent work in mathematical logic has suggested one possibility of which, I believe, Myhill would have approved (personal communication). It has suggested that a full characterization of musical cognition may require what have been called *trial-and-error* processes by Putnam (1965) and *limiting-computable* processes by Gold (1965). Such processes can be characterized quite precisely, and they can be carried out by digital computers, but they can do things that computations cannot. For this article I will take Myhill's thesis to be making a more precise claim in terms of such processes—the claim that, theoretically, musical thinking cannot be fully characterized in terms of computing procedures, but that it probably can be fully characterized in terms of the slightly more powerful trial-and-error procedures.

What Myhill's Thesis Implies for Cognitive Musicology

To say that we cannot fully characterize musical thinking in computational terms need not imply that we cannot fully characterize musical thinking in mechanical terms. It simply says that, if we hope to do so, we will have to do it in terms of machines that are more powerful than computing machines. That is not so different from the claim that, if we want to characterize the physical world numerically, we will have to use numbers that are more powerful than the rational numbers. It is only a theoretical claim, but it has some practical implications, too.

We tend to forget that today's cognitive science is itself an extension of the conceptual toolkit of the behavioral psychologists, who also feared that any such extension would be unscientific. Recall that, earlier in this century, psychologists such as B. F. Skinner (1953) suggested that all human cognition could be characterized in terms of associative networks alone. Allowing anything more seemed to them to be *mentalism*, and therefore could not be considered scientific. Then Chomsky (1957; 1959) suggested that, for theoretical purposes, the machinery of associative networks was not enough. Chomsky claimed that a full account of the mind's ability to handle the grammar of natural languages requires at least some of the more powerful machinery of the computing machines. Just as Chomsky claimed that the machinery of associative networks (of what we now call the *finite automata*) is not enough to account for all linguistic thinking, Myhill claimed that the machinery of the computing machine is not enough to account for all of musical thinking.

Fig. 1. The nine dots of the "connect-the-dots" problem.

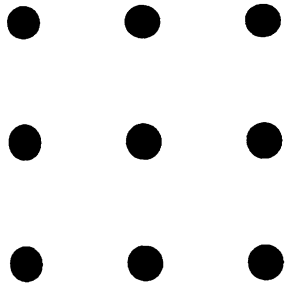
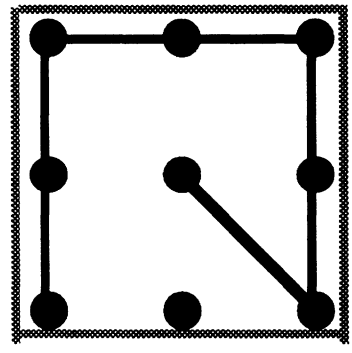


Fig. 2. A typical incorrect solution.



If this claim is true, then people who are trying to characterize all musical cognition in computational terms are doing what people often do when they are first given the following problem: Draw four straight lines through all the dots shown in Fig. 1 without lifting your pencil from the paper.

Typically, when people first see this problem, they assume that their lines have to stay within an imaginary "box" drawn around the dots. So, when they try to solve it, they find themselves forced to leave at least one dot out, giving them something like the incorrect solution shown in Fig. 2.

It is only when they realize that they can go outside the limits of their imaginary box that they find a correct solution, such as the one shown in Fig. 3.

Myhill's thesis claims that people who limit themselves to computational accounts of musical cognition are doing something very much like this. They are unnecessarily restricting what they allow themselves to do in ways that are not forced on them by the nature of the problem. Since this may be preventing them from solving at least some of their problems, it might be worth their while to think about drawing at least some of their lines outside the computational box.

It's Not All Darkness Outside the Computational Box

One of the reasons many people studying musical cognition feel that they have to stay within a computational box is that they consider it possible to find a full account of musical cognition within it. After all, cannot anything that that brain does be simulated, to any degree of accuracy we like, by the digital computer? I happen to believe—and I think that Myhill believed—that it can. But to say that a computer can do something is not the same as saying that it can be done by computing. Despite their name, computers can do more than compute.

To see how this might be, consider what mathematicians call the *halting problem.* The halting problem asks you to write a computer program (call it *Halts*) that, given any program P and input i, will effectively (i.e., computably) determine whether or not $P(i)$ (P applied to the input i) will halt. The black box description of this problem is illustrated in Fig. 4.

Mathematicians have proved that this problem cannot be solved by a computing machine because no computing program can do what the Halts program is supposed to do. But it is unsolvable only if you require the program that solves it (Halts) to have all the features of a computation. If you are willing to give up just one of those conditions—what we might call the *announcement condition*—then you can program a computer to solve this problem in spite of its purported unsolvability. People unfamiliar with mathematics seldom realize that when mathematicians prove a problem unsolvable, they almost always only prove it unsolvable by specific methods. The program that solves it is both digital and deterministic, as a computation is. It is controlled by a finite program and produces its results using finite space in finite time—like a computation. What it lacks is the ability to tell us when it has found its final result.

Here is how such a noncomputing procedure operates. You give it a program P, and an input i, ask-

Fig. 3. A correct solution that "goes outside of the box.

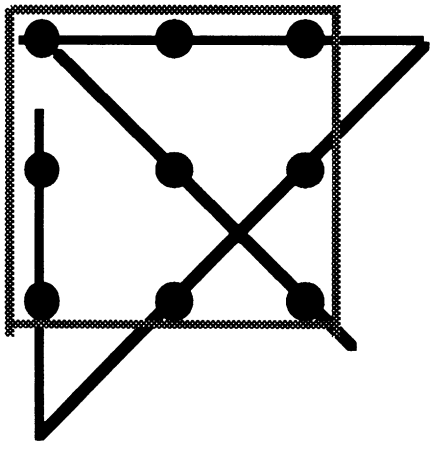
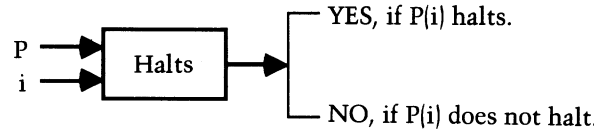


Fig. 4. What the Halt *program—which solves the halting problem—does.*



ing it to determine whether or not *P*, running on *i*, will halt. It works on your problem for a while and generates an answer. That is not necessarily its final answer because it may at any time "change its mind" and produce a different answer. You know that it will eventually give the right answer and that once this has happened, it will not change its mind again. But it will not announce when this has happened.

That may seem like a rather minor matter but, as I will argue, it is not. For practical purposes, it is not minor because you cannot take such a procedure's result as final and shut the machine off. Theoretically, it is not minor because as soon as one gives up the announcement condition there are things one can do that one cannot do with computations alone. For example, the following procedure, which meets all the conditions of a computation except the announcement condition, "solves" the halting problem. Given *P* and *i*, let the procedure start by printing NO. Then let it simulate the operation of *P* on *i* step by step. Turning (1936) proved that such a simulation always exists. If, at any moment, the simulation halts, let our procedure erase the NO, replace it with a YES, and halt.

It is not hard to convince oneself that the result (YES or NO) that this procedure produces is always right and that it is always produced deterministically, in finite time using finite space. But since Turing proved that the halting problem cannot be solved by a computation, this cannot be a computation. What it lacks is the fact that it does not satisfy the announcement condition. Its final result is correct, but it cannot announce its correctness in general. It has to "keep an open mind." That is how a trial-and-error procedure differs from a computation. What Myhill's thesis tells us is that musical thinking requires keeping a more open mind than computations allow.

Musical thinking is not the only kind of thinking that might require such nonannouncement models. Thus Popper's (1955) account of scientific thinking suggests that we need such models to characterize scientific thinking. By Popper's account, scientists behave according to our procedure to solve the halting problem. They may get the correct account of some aspect of nature in finite time, but they cannot know when their account is finally correct. By Popper's account, any theory that a scientist puts forth is always tentative. Further evidence can always disprove it. Science can always change its mind.

The claim that scientific thinking can be fully represented in strictly computational terms is the claim that the scientist can always announce—perhaps with Archimedes's traditional "Eureka!"—when he or she has found the truth. Popper argues that the claim that this can always be done by the scientist misrepresents the scientific method, whose results must, by definition, remain always tentative and open to challenge. Myhill argues that the claim that musical thinking can be fully represented by computations alone misrepresents the nature of musical thinking.

Let me try to be a bit more precise about what Myhill's claim really involves. When a computer carries out a computation, it must meet the announcement condition. Internally, it may change its mind as often as it likes, entertaining any number of hypotheses and subsequently rejecting them. But it must eventually not only stop changing its

mind and generate a result, but also tell us when it has produced its final result so that we can take the result away with us and turn the machine off. A trial-and-error machine need not meet this condition. You cannot tell the difference between a computing machine and a trial-and-error machine by watching them work. The difference between them is not in what they do. They both operate in precisely the same way. The difference between them lies in how we interpret what they do—in what aspect of their behavior we think of as defining their result. An easy way to put it is as follows. When we think of a computer as a computing machine (limited to doing computations), we count its first output as its result. When we think of it as a trial-and-error machine, we count its last output as its result.

Myhill's thesis says that we need trial-and-error machines to fully characterize musical thinking. People to whom I have put this proposal have two reactions. Some yawn, and tell me it is trivial. Computer programs that change their minds are commonplace in artificial intelligence, so trial-and-error and error procedures are nothing new. Others put on a pained look and tell me it is impossible. Human beings cannot keep their minds forever open. Scientists have to publish their theories eventually and composers have to publish scores.

These objections miss the point. It is true that computing procedures can change their minds. Programs that use backtracking, such as the composing programs of Gill (1963), change their minds. They make one decision and, when that decision fails to give satisfactory results later on, they back up and change that decision. Programs that use heuristics (Ames 1987a) also change their minds internally, replacing a good decision made at one time with a better one made later on.

But, if such programs are to remain computing programs, there must come a time when they stop changing their minds and generate a final result. At this point they print out their result—perhaps a piano sonata—and stop. That is what trial-and-error procedures do not have to do. Theoretically—but not practically—we can think of them as running forever, much as we can think of the decimal expansions of irrational numbers, like π and $\sqrt{2}$, as being expressed by infinitely long decimal numbers.

Competence Models

Just as we cannot write out the full decimal expansion of an irrational number in practice, we cannot do musical thinking forever. To represent musical thinking by a process that runs forever is a theoretical idea and it needs to be sharply distinguished from practical attempts to simulate musical thinking. For this purpose, a distinction due to Chomsky is particularly apt.

Those who try to write computer programs that duplicate the results of human musical thinking are trying to develop what Chomsky (1965) called a *performance model.* Notice that the term *performance* is being used here in an extension of its musical sense. A performance model does not just model the way a pianist performs the act of playing a Beethoven sonata. It can model the way a listener performs the act of listening to such sonatas, or the way a composer performs the act of composing them. Myhill's thesis concerns what Chomsky called a *competence model.* There are sharp differences between performance models and competence models that one must understand before one can understand what Myhill's thesis is all about.

Let us start off with two differences between competence and performance models. First of all, a performance model tries to characterize what we *do* whereas a competence model tries to characterize what we *know.* Second, a performance model is practical. A competence model is strictly theoretical.

A performance model of the human ability to deal with natural languages, for example, takes into account the practical limitations of what the human mind can handle. If it is a performance model of a speaker, it need only generate sentences of reasonable length, and thus only a finite number of sentences. A competence model, on the other hand, is intended to capture what a native speaker knows and what he or she could generate in principle. It is not limited to representing what such a speaker can do, in practice. Competence models characterize what a speaker might do in principle. In principle, a speaker might generate sentences of arbitrary length and, therefore, infinite sets of sentences. Chomsky's competence models of grammatical English generate arbitrarily long sentences, and thus

infinitely many of them, even though no human will ever produce infinitely many sentences.

We can distinguish two kinds of performance models of human musical thinking—a *practical model* and a *psychological model.* For example, a practical model of the process by which sonatas are composed is intended to give you a computer program that will produce sonatas for you. To do this, such programs must be efficient enough to run on existing computers, using reasonable amounts of time and space. They demonstrate their adequacy by producing one or more suitable sonatas within such practical limitations.

Practical models always meet the announcement condition. If you want to use a computer to compose a piano sonata, you want it to tell you when it has finished so you can mail off the score. What you do not want in such circumstances is to have the computer leave you hanging. You do not want it to say to you, in effect, "Do not take my sonata yet because I may later change my mind about how I want it to go."

Psychological models differ from practical models in that they not only try to get the job done, but they also try to get it done in somewhat the same way that people do it. A psychological model of the way that Beethoven composed piano sonatas would try not only to produce some Beethoven-like sonatas, but also to do it in the manner in which Beethoven did it. It might try to produce drafts like those Beethoven produced, or it might try to characterize other aspects of Beethoven's composing—the errors he made, or the order in which he wrote different portions of a given work. Psychological models are intended to contribute to the psychological study of human musical thinking. Like practical models, phsychological models have to satisfy the announcement condition. They cannot, in this practical world, run forever.

It is only when you are talking about musical thinking theoretically that competence models make sense. Competence models of musical thinking attempt to characterize what a composer knows by characterizing, not what he or she can do in fact, but what he or she can do in principle. Looking at it theoretically, what Beethoven knew about writing piano sonatas could have been used to produce an infinity of different sonatas, much as what human beings know about the grammar of their native languages can be used, theoretically, to generate an infinity of sentences. Nobody pretends that human beings can actually produce infinitely many anythings. But ignoring the finite limits of human abilities helps us get at the underlying structure of human thinking.

Because, when we talk about competence, we are interested in what a system can do in principle, rather than in fact, we ignore the time things take or the space they require. At most we insist that they take finite amounts of time and space. That is why programming considerations, such as the time saved by using heuristics or backtracking, disappear. That is why hardware considerations, such as the time savings produced by using parallel architectures or neural nets, are irrelevant.

Such idealized and ultimately unrealistic theoretical models of the world play a central role in many sciences. Newtonian physics, for example, begins by studying motion under highly idealized conditions, with objects moving in the absence of friction or gravitational fields and with their entire mass concentrated in a single point. Such motion is strictly theoretical and it is a good place to start.

One reason that we use such idealized theoretical models is that they give us a good way to break up the job of trying to understand the world. We can thus think of cognitive science as developing its models in three stages. First, someone demonstrates the feasibility of a particular kind of model by developing a competence model—a model that produces the desired results if we ignore practical considerations. That shows that in principle at least the components used in the idealized model are sufficient for the job at hand. Once this has been established, we can work on practical considerations—on making the model efficient enough so that its use becomes practical. It is at this stage that we might want to use parallel architectures, heuristics, backtracking, etc. Finally, if we are psychologists, we might want to change our model so that it not only does what people can do, but does it in the same way that people do it.

Each of these types of models does something different for us. A practical model does something

useful. It might produce Beethoven-like sonatas that Beethoven never wrote. A psychological model does something psychologically relevant. It might tell us something about Beethoven's way of thinking. A competence model does something different from both. It tries to capture, not what Beethoven did or how he did it, but what Beethoven might have done theoretically. It tries to model what Beethoven would have done under idealized conditions, much as Newton's laws of motion try to model what a soccer ball would do under idealized conditions. The practical considerations can always be added later, once the theoretical aspects have been better understood.

Competence models ignore practical limitations in order to try to get at other deeper and more structural aspects of the mind. In terms of them, we can ask such questions as: "Is the machinery of the finite automaton strong enough to account for what the human mind can do?" or: "Is the machinery of the computing machine strong enough to account for what the human mind can do?" Chomsky answered no to the first of these questions. Myhill answered no to the second. Chomsky's answer changed the ground rules of theoretical linguistics. Myhill's answer seems to me to change the ground rules of cognitive musicology.

Three Levels of Musical Thinking

The preceding discussion has tried to clarify what it is that Myhill's thesis says. Now it is time to ask whether its claims are true. Why would a full account of musical cognition require trial-and-error as well as computing machines? According to Myhill, the problems with which musical thinking deals can be split into three categories: (1) those that can be solved *effectively* by means of a total computation, (2) those that can only be solved *constructively* by means of a partial computation, and (3) those that can only be solved *prospectively* by means of "uncomputations."

Problems are said to be solvable by *effective* means if they can be solved by a process that both computes a solution, if one exists, and produces an answer of NO if no solution is possible. Thus, Myhill suggested that the problem of determining whether a group of notes is consonant in the traditional sense is totally computable. There is a single computing procedure that, given any set of notes, will produce an answer of YES if the set is consonant and NO if it is not.

Problems are said to be solvable by *constructive* means if they can be solved by a process that computes the solution, if one exists, but does not necessarily tell us "effectively" when no solution exists. To see how such a problem might arise in musical thinking, imagine that we have an effective (or computable) procedure for composing piano sonatas in the style of Beethoven. Using this procedure, we can answer the question constructively, "Would Beethoven (if he had lived forever) ever have used a particular chord in one of his sonatas?" To do this, we build a constructive (or partially computable) program out of the given effective (or totally computable) program by using it as a subroutine. We let our main program run the subroutine to produce the infinite number of piano sonatas that Beethoven might have written if he had lived forever. As sonatas are generated, we check them to see if the chord we are looking for has been used. If it has, the main program stops and prints YES, but if it has not appeared, it goes on to the next sonata. As soon as a chord has been used, the program knows it and can print YES. But there may be no general way of determining when the program has looked at enough sonatas to be able to say NO with finality.

Partial computations can be thought of as procedures that are—in some sense—half computable and half not computable. Their YES side is computable, but their NO side is not. Myhill's thesis says that there are musical problems whose YES side is not computable and he called such problems *prospective.* A problem is prospective if its YES solutions are also not computable. To produce an example of a prospective problem, one only has to turn the halting problem upside down—to ask for a procedure that computes YES if and only if $P(i)$ fails to halt. No computation can correctly identify all such nonhalting runs of programs. (Again, that is a theorem.)

Prospective Problems in Musical Realms

Problems that are only prospectively solvable exist, but that the question still arises: "Do such processes occur in musical thinking?" Myhill suggested that they do and, as an example, he gave the process of identifying a piece of music as beautiful. Let me try to argue his claim with two slightly different examples: the problem of trying to generate all the piano sonatas that Beethoven might have written and the problem of characterizing the set of all classical styles.

What Beethoven Knew

One reason to try to construct a program that will generate all the piano sonatas Beethoven might possibly have written is to try to characterize what Beethoven knew about such sonatas. How might we proceed? Assume that piano sonatas are written a measure at a time—which is probably not true, but it is a way to get started. We need an algorithm to write the measures of our sonatas, one after the other. To see how such a procedure might work, let us start twenty measures into a sonata and ask ourselves how our algorithm is going to choose the next measure. My concern here is not with the exact procedure, but rather with trying to determine what kind of procedure we can use—effective, constructive, or prospective.

Assume that Beethoven could tell by listening whether or not a measure is suitable in position twenty-one, given the preceding twenty measures. In other words, assume that Beethoven had an effective procedure that—given the first $n + 1$ measures—could compute YES if a given measure were appropriate in the next position and NO if it were not. Assume, also, that Beethoven had an effective procedure to determine when a piece was finished. (Beethoven seems to have had some difficulties with this.) I claim that if such a computing procedure exists, then an effective composing procedure that generates all the possible piano sonatas that Beethoven could have written exists as well.

I know this because I know how to write a generating program that uses this recognizing program as a subroutine. I would start my program by writing all measures that could appear in position one. I would have it try out all possible starting measures—using the recognizing procedure I am assuming to exist on a sequence of length zero. When it has produced all possible first measures, it would put all these pieces of length one on a list. Then it would go through its list, using the same recognizing procedure on all possible measures, to produce a—probably bigger—list of all sonata openings of length two. It would keep going like this, keeping unfinished sonatas on its list until it recognized one as finished. When this happened, the finished sonata would be output and removed from the list. Since there are many, but only finitely many, measures to try at any spot, the program will recognize when it gets to an unfinished sonata for which no acceptable conclusion is possible. That sonata beginning can then be dropped from its list of candidates.

It is not hard to see that this procedure (I have left out a number of bookkeeping details) will generate all the sonatas that Beethoven could have written and thus characterize his compositional style constructively. It is impractical, however: it is too slow and requires too much space.

To turn it into a practical method, we might want to use programming techniques, such as heuristics and backtracking, or hardware ideas, such as parallel processing or neural nets. But none of these techniques would increase the procedure's theoretical capabilities. They simply make it useful.

But we are doing theory; making it useful is somebody else's job. If clever programming and innovative computer architectures cannot get us beyond computing, what can? Whenever we have a situation in which the composition of a whole piece involves a series of steps, each of which involves only the consideration of a finite number of alternatives, and each of which can be computably accepted or rejected, we have a composing procedure that is at least partially computable.

Not all imaginable ways of composing have these properties. Consider a slightly different way to compose piano sonatas. Suppose there is an effective way to produce acceptable sonatas, but that Beethoven was trying to do more than produce acceptable

sonatas. Suppose that he was trying to write one of the 100 best sonatas that could possibly be written in his general style. To do that, he would have needed a way to rate the quality of the sonatas. Assume that he had an effective rating procedure that rated sonatas on a scale of one to ten (with ten being the highest rating). Since there can be more than 100 sonatas tied for the highest possible rating, let us agree that if this should happen, any of those more-than-100 best will be an acceptable composition.

Now all we have to do is to find one of the 100 best possible sonatas in the style that our computing procedure generates. One way to do that is to generate all the acceptable sonatas, one at a time. Each time it generates a sonata, our evaluating procedure can compute a measure of its "goodness" on a scale of one through ten, putting the first 100 it generates on a list. Then, as it produces more sonatas, each one could be checked to see if it rates higher than the worst one already on the list. If it does, then the new sonata would replace the worst one so far on the list. At any moment, the list will contain the 100 best sonatas produced so far. When can our procedure stop and output one of the hundred best sonatas, knowing that it will be among the 100 all-time best? If our procedure finds a sonata with a perfect rating (ten), it can output that sonata knowing that it must at least tie for a place among the 100 or more all-time best. But suppose that no such sonata is encountered. When can it stop and output a sonata with a rating of nine? Well, it cannot, unless it can figure out some way to be sure that there are not 100 ten's still ahead of it. What it can do is to put out a nine-rated sonata after some period of time and hope that 100 tens do not come along. The nine-rated sonata is a tentative output. If fewer than 100 tens come along later, it will remain an acceptable composition.

Notice what happens when this procedure tries to generate the sonatas in this set of what we might call "Beethoven's 100 Greatest Hits." It will put out a nine-rated sonata as a candidate for membership in this set in finite time. What it cannot do is announce in finite time that it has definitely found a sonata that meets its requirements. What we can say is that it will find a 100 such sonatas in finite time. At some point, it will have a list of a 100 sonatas that will not change. But we may not know when that time has come.

You might say that is ridiculous as an account of how to compose piano sonatas. There has to be some finite limit on the length of such sonatas. No sonata can take a day or more to play. So after the procedure has gone through all the sonatas that take less then a day to play, it can produce the 100 best–so–far and stop. If we say that, however, we are missing the point. If we want to get at the structure of musical thinking at this level, we have to allow sonatas of arbitrary length, much as Chomsky allowed sentences of arbitrary length to count as grammatical. If he had done otherwise, the distinction between, for example, a context-free grammar and a regular grammar would have collapsed. His argument against behaviorism would have collapsed too.

My choice of Beethoven's compositional method as an example here is not coincidental. I think that this kind of trial-and-error account does offer a basis for a plausible account of how Beethoven composed. But the utility of the trial-and-error mode of modelling musical thinking does not depend wholly on this one example. Different trial-and-error models of composing might be more plausible. Trial-and-error accounts might provide the basis for better accounts of how composers' styles develop than strictly computational accounts can. Interpretations of a given piece, either from the viewpoint of the listener, or of the performer, seem also to have this always-tentative character that trial-and-error models capture.

It therefore seems to make sense to at least think about allowing such trial-and-error accounts to be used in giving accounts of musical thinking. If they are not needed for a full account of musical thinking, then that would mean that musical thinking comes to more final, more certain, conclusions than scientific thinking can, or, to put it in another way, that musical thinking requires weaker abstract machinery than scientific thinking does. That is not impossible, but it strikes me as somewhat implausible.

What difference does it make if we expand our conceptual toolkit to include trial-and-error ma-

chines as well as computing machines? It gives us more powerful conceptual tools, and that can make a difference. Consider one difference that allowing the irrational numbers into the conceptual toolkit of the physical scientist made. Mathematicians think of the value of an irrational number, like $\sqrt{2}$, as the result that its increasingly more accurate decimal expansion (to more and more decimal places) approaches "in the limit." Each decimal expansion gets closer and closer to the right value, but none is exactly right. Adding the "end" of this process, which takes infinitely long to "complete," does nothing to change the way the world is. But it does change the way that we think about the world. It adds the irrational numbers to the set of tools we can think with and that can make our thinking both simpler and more powerful. It changed the way we think about many things, and it may have changed the way we think about music. It may have made it possible to discover the notion of equal temperment.

Recall that the Pythagoreans developed a system of tuning based on rational (fractional) intervals that worked well enough for some purposes, but made certain chords sound slightly off, and limited the ability of a performer to move freely between keys without pausing to retune. Well-tempered tuning handles these problems well enough to make modulation to different keys possible without retuning and thus to make the classical style of composition possible.

But it is possible that, if we had not had the irrational numbers, nobody would have ever thought of equal temperment because the theory behind it uses an irrational number—the twelfth root of two, or $\sqrt[12]{2}$—in a fundamental way. When you determine the frequency of the notes in the chromatic scale to adjust the tuning of an instrument to equal temperment, you start with the frequency of the first note in the chromatic scale, f Hz. From this, you compute the frequency of the next note by multiplying f by $(1 + \sqrt[12]{2})$. You keep multiplying the frequency of the current note by $(1 + \sqrt[12]{2})$ to get the frequency of the next note on the chromatic scale. The crucial point for our purposes is that this might be very hard to dream up if you did not have irrational numbers, like $\sqrt[12]{2}$, to think with.

Notice also that—for practical purposes—the irrationality of $\sqrt[12]{2}$ in the result makes no difference. To tune a string to what the human ear perceives to be the value of the frequency $(f^* (1 + \sqrt[12]{2}))$ a rational approximation—indeed one to the nearest whole number—will suffice. The human ear cannot tell when a frequency is off by only a fraction of a Hz. Where the irrational numbers come in is in the thinking that tells us that $f^* (1 + \sqrt[12]{2})$ is the right value to approximate; it disappears in the practical version of the result.

The effect of adding trial-and-error procedures to the conceptual toolkit we use to think about musical cognition could be similar to what adding irrational numbers to our conceptual toolkit might have done for our thinking about tuning. It might have made that thinking more powerful, and thus allowed us to discover new ideas. Who would have expected the concept of the irrational number to lead to a good way to tune the instruments of an orchestra? Who knows what the use of uncomputable models in cognitive musicology might lead to? Myhill's thesis suggests that it might be worth trying to find out.

Adding trial-and-error procedures to the toolkit of the cognitive musicologists might also change the kinds of questions they ask. I think that the questions scientists ask are as crucial as the answers they give, a view that is at least partially a result of my reading of Kuhn (1962). That is one of the things that the study of abstract, rather than practical, models can do. It can change the questions we ask. Consider, for example, what happened when physics changed from Aristotle's abstract theory of motion to Newton's. According to Aristotle, an object in motion comes to a halt unless something keeps pushing it. It is easy to conclude that Aristotle was right. Try kicking a soccer ball and you will see that it will stop unless you keep kicking it. Aristotle's theory of motion led to the question: "What keeps the planet Venus moving?" and the generally accepted answer was: "The Prime Mover." According to Newton's abstract theory of motion, an object in motion stays in motion unless something stops it. I call this theory abstract because there is almost nothing in the motions we observe on earth that supports it. Now that we accept it, we no longer

ask why Venus moves, but rather why soccer balls stop. The generally accepted answer is "friction." The change in the question changed physics for the better.

The trial-and-error model of composition could produce a similar "question shift" in cognitive musicology. Instead of asking why a composer keeps revising a work, we now might ask why he or she ever stops. By this account, Mozart may have been the oddball and Beethoven the normal composer. By popular accounts, which are of doubtful authenticity (Sloboda 1985), Mozart produced his works in one fell swoop. Beethoven, on the other hand, kept revising and revising. The trial-and-error account of composition makes us ask, "Why did Mozart stop?" Two possible answers are: "Because he had lower standards," or "Because he was more efficient at generating, presumably unconsciously, lots of good alternatives."

The trial-and-error model could also have much the same effect that the introduction of the limiting process had in physics. That introduction led to the calculus which made it much easier for physics to study motion. Perhaps adding trial-and-error models to our conceptual tool-kit will allow us to focus better on "motion" in musical thinking—how musical thinking, and its products, grow over time. Finally, the adoption of a trial-and-error model might refocus our attention on the role of competence models in musicology and on the knowledge that underlies musical thinking—rather than just the performance that evidences that knowledge—much as Chomsky's introduction of more powerful competence models in linguistics has refocussed that science on competence models and linguistic knowledge. These seem to me to be good reasons for at least considering more the powerful trial-and-error models of musical thinking.

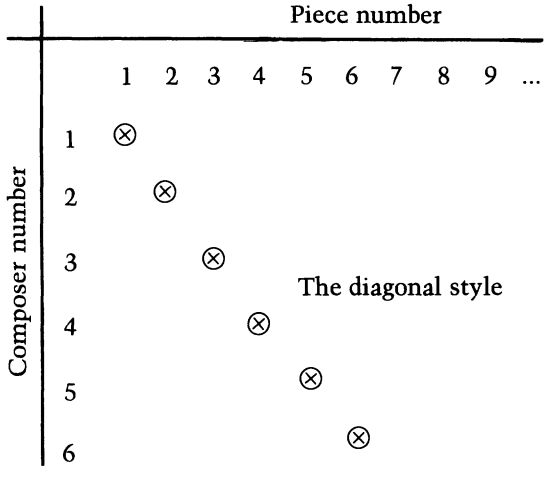
Characterizing the Classical Style

Cognitive musicology can look at musical products as well as musical processes. Consider, for example, the set of all "beautiful" music that Myhill suggested—in effect—could not be characterized by computation means. How might a computation characterize the concept "beautiful" in music? There are two ways. Given an object, a program could determine a concept *effectively* if it can determine by a computation whether or not the object falls under the concept. If such a program were to try to characterize the beautiful in music, it would take a score as input and produce an answer of YES if the score were that of a beautiful piece of music or NO if it were not. A program could also determine a concept *constructively* if it could generate a list of all the things that fall under the concept. To characterize beautiful in music, such a program might generate the beautiful musical scores. Myhill suggested that the set of all beautiful things could not be characterized in either of these ways by a computer program. Let me use a related but different example of a problem with this property—the problem of trying to define all possible personal styles in the classical tradition.

If we were to define what it means for a composer's style to fall within the classical style for students in an introductory music appreciation course, we might try to come up with a list of features that all compositions in this style had to share. We might then tell our class that a composer wrote in the classical style if and only if all his or her pieces had those features. If the presence of these features can be determined effectively, then such a set of features defines the concept of a classical style effectively.

For a more sophisticated audience, we might not be able to find a fully adequate list of necessary and sufficient conditions. The fact that concepts are often hard to define in terms of necessary and sufficient conditions has been much discussed in philosophy as a result of the analysis of Wittgenstein (1953), and in psychology as the result of the analysis of Rosch (1978). Lacking such a set, we might try to determine the classical style constructively as follows. We write a program P that generates a series of programs $C_1, C_2, \ldots$, such that each one of the C_i, represents a different composer. Program C_i then generates the compositions of composer i. A composer could then be said to write in the classical style if and only if there were some i such that his or her compositions were generated by C_i. Assume that there are infinitely many such com-

Fig. 5. The diagonal composition style.



posers because a good general style should allow for an unlimited number of individual styles within it. Our master program generates a sequence of programs C_1, C_2, . . . , each of which generates an infinity of pieces, with C_i generating all the pieces in the style of composer number i. Under quite reasonable assumptions, we can argue that no such characterization of all classical styles can be successful because it has to leave out some possible classical composer. Consider for example what we might call the *diagonal composer.* This composer's style, $C_{diagonal}$, is constructed by first laying out all possible classical composers in rows of a table, as shown in Fig. 5, and then "diagonalizing" the table as follows.

We take the first piece of composer one and change it a bit, making sure that the changes keep it in the classical style. Then we take piece two of composer two and change it so that it also stays within the classical style. In general, we change piece n for each composer, C_n for n=1,2,3,4,5, . . . , in some standard way, making sure that our changes keep the new pieces classical in style. The resulting diagonal style is the style of composer $C_{diagonal}$. It is wholly classical, but is not on the list of all personal styles that we wanted to count as classical styles. It differs from the style of composer number n in its n-th piece, at least. Since the procedure we used was based on finding any possible computable procedure for generating all styles, it follows that no computable procedure of this general form can list all personal styles of classical composers. This looks like a trick, and what is more, it seems to find only one exception, which does not seem so important, but this form of argument is widely used in mathematics, and it is considerably more convincing than it may seem at first.

More generally, whenever we have a set of objects, any of whose computational enumerations can be "diagonalized" in this way, we have a set whose members cannot be characterized (or generated) by a computation because, given any computable enumeration, we can always generate an element in the set that any candidate computation would have left out. Diagonalization is a second way that we can show that something cannot be characterized by means of computations alone and that something more powerful, like a trial-and-error process, might be required. If we find that "beautiful" cannot be characterized computationally, this also says something about the process of listening to music. If "beautiful" in music cannot be defined computationally but we can recognize it by listening, then the process of listening must involve more than computing.

Uncomputable models developed from trial-and-error procedures emphasize the adaptability of musical knowledge. Uncomputable models developed by diagonalization seem to emphasize the role of self-awareness in musical thinking. For example, an awareness of what is involved in a given style as a whole may be necessary to allow a composer to go beyond that style to develop a new style. One of the things that made Mozart a great composer in the classical style is not just that his pieces were good, but that he generated a new personal style. Surely this depended on knowing how the classical style of his time had been developed by others, and then going beyond what they had done. One way to create a style that is basically different from any other is to use such a diagonal process. R. Kirsch (personal communication) has suggested that some new styles in the visual arts seem to have been developed by an almost conscious application of such a diagonalizing procedure. Composers need not be conscious of using this diagonalization process for it to provide a good theoretical description of what they

are doing when they develop a new style. If the idea of a "good" style is one that can always be expanded by such diagonalization, we have another reason to believe that a full characterization of musical thinking will require more than computations.

If this is so, it may have some interesting implications that can be stated informally. Myhill saw the prospective (uncomputable) nature of the concept of beauty as implying "that there exists no formula or attitude, such as that in which for example the romantic believed, which can be counted upon, even in a hypothetical infinitely protracted lifetime, to create all the beauty that there is" (Myhill 1952). He also suggested that his analysis led generally to an analog of Gödel's celebrated incompleteness theorem in aesthetics, which states that "there is no school of art which permits the production of all beauty and excludes the production of all ugliness," and to an analog of Church's (and Turing's) celebrated theorem about the undecidability of Hilbert's *Entscheidungsproblem* (decidability problem) that "there is no token (as pleasure or the like) by which you shall know the beautiful when you see it."

In terms of our Beethoven example, the analog of Gödel's incompleteness theorem is that there is no single (computable) technique that will allow one to compose all possible classical piano sonatas, even if one ignores practical limitations of time and space. The analog of Church's theorem is that you will have to tell your music appreciation class—if you are honest with them—that you simply cannot give them a wholly adequate definition of the classical style that they can then apply effectively on the final exam.

Conclusion

At the beginning of this article, I suggested that many people prefer to stay inside the computational "box" in developing cognitive theories because they fear the wilderness that they believe lies outside of it. I suggested that their fear might be unwarranted. I can now be a bit more precise about this statement. One concept that makes the territory outside the computable box viable for those who like their theories precise is the notion of the trial-and-error procedure. It shows that some uncomputable processes can be characterized with precision. Problems inside the computational box are problems whose solutions require only what Myhill called *technique.* Solving problems that are outside of it requires what he called *insight.* A problem requires only technique if it can be solved by the thoughtless application of a recipe (or algorithm). It requires insight if you have to understand it to solve it. Notice that this difference can be relative to what you know. Before you learned the algorithm for doing long division, dividing required insight on your part and there were lots of division problems that you could not solve. Once you learned the algorithm for doing long division, it no longer required insight, and you could divide any numbers the teacher gave you (unless you made careless errors or ran out of time). Myhill's thesis suggests that in some cases, this difference is absolute. Although many people seem to believe that every problem can be solved by a technique, our failure to find techniques to solve some problems that arise in musical thinking may not be, as Myhill put it, "merely an accidental gap in knowledge, which will be closed in the next few thousand years if death and dynamite do not intervene." There are some problems whose solution simply cannot be reduced to technique, no matter how hard we try. Some of these are found in scientific thinking. Myhill's thesis suggests that others are found in musical thinking.

Do not get me wrong. I have nothing against technique. Once we find a technique for doing something, it can be encapsulated and stabilized. It can be known with finality and conveyed with rigor. In some sense, techniques are ultimately static and fixed. This is what is good about them, but it is also what is bad about them. They are rigid and fixed. Modern physics gets much of its power from its ability to deal not just with objects standing still—such as Archimedes in the bathtub—but also with moving objects—such as the things Galileo rolled down inclined planes. It is interesting that this power comes in large part from calculus, built on the notion of passing to a limit. This is not unlike the notion that gave rise to the trail-and-error

process, as Gold (1965) observed. Computational processes can account for many aspects of musical cognition, and they should almost certainly be used to develop such accounts whenever possible. Myhill's thesis claims, however, that it is not always possible.

Trying to characterize all musical cognition in terms of computations alone may be a bit like trying to paint all landscapes without using green. It can be done, but the results will sometimes seem to be slightly off, particularly when the landscapes being portrayed contain a lot of grass. Myhill's thesis tells us that there is something like grass in the musical landscape and that we therefore ought to add uncomputable processes to our conceptual palette. What it suggests is not unlike what Einstein suggested when he said, "A scientific explanation should be as simple as possible, but not simpler."

Acknowledgments

An earlier version of this paper was presented at the AAAI First Workshop on AI and Music, St. Paul, Minnesota, 24 August 1988. I want to thank Charles Ames and Otto E. Laske for helpful suggestions. They should, of course, be granted the customary absolutions.

References

Ames, C. 1987a. "Music, AI in." In S. Shapiro, ed. *Encyclopedia of Artificial Intelligence.* New York: Wiley, pp. 638–642.

Ames, C. 1987b. "Automated Composition in Retrospect." *Leonardo* 20(2): 169–185.

Chomsky, N. 1957. *Syntactic Structures.* The Hague: Mouton.

Chomsky, N. 1959. "Review of Skinner's *Verbal Behavior.*" *Language* 35(1): 26–58.

Chomsky, N. 1965. *Aspects of the Theory Syntax.* Cambridge, Massachusetts: MIT Press.

Gill, S. 1963. "A Technique for the Composition of Music in a Computer." *The Computer Journal* 6(2): 129–133.

Gold, E. M. 1965. "Limiting Recursion." *Journal of Symbolic Logic* 30(1): 28–48.

Kuhn, T. S. 1962. *The Structure of Scientific Revolutions.* Chicago: University of Chicago Press.

Laske, O. E. 1988. "Introduction to Cognitive Musicology." *Computer Music Journal* 12(1): 43–57.

Myhill, J. 1952. "Some Philosophical Implications of Mathematical Logic: Three Classes of Ideas." *Review of Metaphysics* 6(2): 165–198.

Popper, K. R. 1955. *The Logic of Scientific Discovery.* New York: Harper & Row.

Putnam, H. 1965. "Trial-and-Error Predicates and the Solution to a Problem of Mostowski." *Journal of Symbolic Logic* 30(1): 49–57.

Rosch, E. 1978. "Principles of Categorization." In E. Rosch and B. B. Lloyd, eds. *Cognition and Categorization.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Skinner, B. F. 1953. *Science and Human Behavior.* New York: Macmillan.

Sloboda, J. A. 1985. *The Musical Mind.* Oxford: Oxford University Press.

Turing, A. M. 1936. "On Computable Numbers, with an Application to the *Entscheidungsproblem.*" *Proceedings of the London Mathematical Society,* ser. 2, 42: 230–265.

Turing, A. M. 1950. "Computing Machinery and Intelligence." *Mind* 59(4): 433–460.

Wittgenstein, L. 1953. *Philosophical Investigations.* New York: Macmillan.