

# THE ELTHAR PROGRAM



BRAD GARTON

**D**URING MY YEARS as a graduate student, I often felt that I was leading a musical 'double life'. By day I was immersed in the music of digital synthesis and signal-processing. In the evening, I would gather with friends to participate in long free-form improvisation sessions. These two activities seemed fundamentally different to me—improvisation gave me a sense of instantaneous freedom and possibility that has been invaluable to my compositional growth; working with computers satisfied my conflicting desire to build and rearrange sounds with the control offered by digital techniques.

Because of the qualitative differences between the two, I had thought that improvisation and computer music composition would always be separated in my life, exerting only indirect influence upon each other. Two realizations moved me towards unifying these activities. Upon examining my

feelings about improvisation, I realized that the foundation for my enjoyment of the improvisation sessions was the interaction with others. The free interplay of ideas among individuals was the substrate underlying the freedom and unboundedness of the sessions. I was also beginning to realize a sense of interaction with some compositional algorithms I was experimenting with on the computer. By creating algorithms which could generate sound synthesis data by following certain rules, I was able to deal with the computer on a higher conceptual level. For example, I wrote an algorithm which would create rhythmic patterns when given probabilities that particular drum sounds would occur at certain points in time. My interaction with the computer shifted from having to specify data for every single drum beat within a time-span to specifying some global characteristics (through probability functions) and letting the computer fill in the details. In essence, I was able to tell the computer "do something like this . . . for thirty seconds," then listen to the output and adjust higher-level control probabilities to reflect my compositional goals.

Through experiments with probabilistic algorithms such as this, I began to build a richer interaction with the computer. I was also beginning to treat the machine as a separate entity, leaving many of the note-to-note decisions in the hands of some algorithm. Of course, these simple interactions were only a pale shadow of the tangled web of relationships among participants in an improvisation. It occurred to me, however, that I might be able to model a more complex interaction using the symbolic processing capabilities of the computer. (The idea of designing a high-level software interface for musical creation is certainly not new. Most computer-music practitioners have felt the need to enter music data at a level beyond simple (and tedious) numeric specification of parameters. Indeed, much of the computer's musical potential cannot be accessed without the capabilities of a powerful interface.)

The Elthar program is the result of this modeling attempt. Elthar is a large signal-processing expert system designed to interpret natural-language requests from the user. Elthar learns how to use digital signal-processing algorithms by observing how they are used. Elthar was also intended to be more than just another tool to be used in the service of external compositional ideas; the program modeled an interaction that was to be an integral part of the creative process. My design of Elthar reflects my belief that the environment in which I create music is a prime determinant of the nature of the resulting music.

This compositional approach differs from a stance I had adopted in the past (one shared by many western composers) that music creation is primarily a *mental* activity—once the music has been conceived in the mind of the composer, it only requires the appropriate conduit to be translated into sound. In fact, this has been one of the big attractions for composers to use



the computer as a musical instrument. The general-purpose nature of the machine means that it can theoretically be programmed to act as whatever sonic translator is needed by the composer to realize his abstract musical ideas. "Any sound you can imagine" has been the motto of many digital synthesis enthusiasts. This view of the creative process largely ignores any effects that the mental-to-physical conduit might have on the actual conceptualization of the music. Working with computers to realize sounds in a variety of different ways led me to see how profoundly the tools and techniques I use do effect my creative imagination.

The effect is much more than the simple notion: "I shall write a violin piece, therefore I imagine the sound of a violin." At the point that I decide to write a violin piece, I invoke a set of prejudices and conventions that establishes the framework for my composing. As I go about the task of writing the violin music, the attitudes and working habits derived from this framework play a major role in the shaping of the piece. Even before I begin to imagine the music, the assumptions about music I make when I adopt the violin-composing stance will set the stage for my imagination, channeling my efforts in certain directions. One of my goals with Elthar was to try to come to terms with some of my basic musical assumptions, to try to codify a set of working methods appropriate to a particular situation.

In this paper I will be addressing three critical issues raised by the Elthar program: (1) Why create such a thing? (2) How does it work? (3) What do you do with it? I want to emphasize that my primary motivation for making Elthar was to create a vehicle for producing music—a vehicle intended to challenge and expand my compositional techniques. I wasn't interested in designing a generalized music interface that would capture many different compositional approaches. Elthar is a very specialized and idiosyncratic program intended to magnify and capitalize upon my own idiosyncratic compositional prejudices.

Probably the most important result of the project was that Elthar could act as enough of an alter ego to provide an interesting foil for my musical ideas. Through improvisation, I have found that working with other individuals to create sound is a fruitful and stimulating way to make music. The unique thing about working with Elthar is that the program is able to mirror my own preconceptions about music. The mirrored reflection is distorted, however, causing me to see certain aspects of those preconceptions that I would normally not notice, or that I would take for granted. The examination of these assumptions has suggested a host of new musical avenues for me to explore.

## WHY CREATE ELTHAR?

Consider the following three scenarios:

1. I am sitting, a guitar cradled on my lap in playing position. I bring my right hand up to the strings and place the fingers of my left hand along the fretboard. I strum the strings slowly, making immediate adjustments with my hands to change the sound I am producing. The adjustments serve to align the sound with my sonic imagination; the feedback between what I want to produce and what I am producing is tangibly direct.
2. I am sitting at a table, pencil in one hand and a piece of music paper resting on the table surface. I draw dots and lines on the paper, hoping to capture in the resulting symbols information that will allow sonic images to come to life. Much later (weeks, months, possibly even years) I will give this paper to a musical performer (a guitar player in this case) who will create sounds corresponding more or less to the symbols I have written. The process of producing sound from the symbolic language is extremely complex, involving the interpretation of each symbol through an agent with an entire lifetime of experience completely separate from my own. The only access that agent has to my sonic imaginings is through the symbols I inscribe on the paper.
3. I am sitting at the keyboard of a computer terminal, entering data by typing numbers and letters. When I am finished, the data will be used by a computer algorithm designed to make the computer synthesize a sound. The connection between the data and the resulting sound is inflexible. The numbers are interpreted in a completely determined manner by the algorithm. Changing one of the numerical values will always affect the resulting sound in the same way. If I choose the data and the algorithm wisely, I can use the computer to create any imagined sound. It is my responsibility, however, to identify the correct computer algorithm and corresponding data to realize my ideal sound (to simulate a guitar I may have chosen the Karplus-Strong "plucked-string" algorithm). Once my choices have been made and the data has been entered, the computer will "crunch the numbers" to create the sound. Later (typically several minutes or hours) I will listen to the result.

Each of these has the potential to produce nearly identical acoustic results. The music I write in each case, however, will be different. When I



pick up the guitar to play, I am picking up all my guitar-playing experiences. My musical creativity will be constrained not only by my physical performance capabilities, but also by the way I think musically when I have a guitar in my hands. This mode of thinking has been shaped by a multitude of social, cultural, and personal factors reflecting my personal history as a guitar player.

Similarly, picking up a pencil to write music for some other guitar player brings another set of musical premises into play. The entire concept of written musical notation in our culture will certainly exert a profound influence upon my musical decision-making. The simple fact that the symbols I write are intended for musical interpretation by another individual changes at a fundamental level the way I think about the music. I am preparing a "script" for a performer to read and interpret. Even the visual aspect of the notes on paper may cause me to see patterns and consider options that might not be apparent in another mode of composing.

Finally, the computer also invokes a particular set of personal conventions when used as a musical tool. Like guitar playing and script writing, these conventions reflect the musical capabilities of the computer and my own computer experience. Included within this domain, however, is a capability that separates the computer-music environment from the others. As a general-purpose machine, the computer can be programmed to perform a wide variety of operations, including operations which alter the way I interact with the machine. This means that not only do I have a new set of tools at my disposal, but also the capability to design new tools (and even the capability to design the capability to design new tools). I can imagine a way to do something and, with a little programming, implement that strategy on the machine.

Putting this idea into practice has dramatic effects on the music I produce with the computer. An example: Consider again scenario three above, the computer-music scenario, only this time suppose that I have written two versions of the plucked-string algorithm, and the only difference between the two is the way in which pitch information is specified. One version accepts pitch data written in octave.pitch-class notation,<sup>1</sup> of the western equal-tempered scale, and the other requires pitch to be given by frequency (cycles per second). The effect that the selection of one or the other of these two digital instruments would have on what I write is apparent to those familiar with musical acoustics. Because of the logarithmic nature of our perception of frequency as pitch, it is difficult to think of the western equal-tempered tuning in terms of absolute frequencies. Conversely, because of compromises made in the design of the equal-tempered scale, perfect intervals found in many nonwestern scales are nearly impossible to represent coherently in octave.pitch-class notation. If I were to use the octave.pitch-class plucked-string program, I could not but think of



music in equal-tempered terms. The direct frequency version also carries much baggage—the conception of a piece in the key of 440Hz is vastly different than the conception of a piece in the key of A major.

Is this simply equivalent to selecting an instrument and then, given the musical assumptions surrounding that instrument, writing a piece for that instrument? I think not, because the implementation of sound-producing algorithms on the computer is under my direct control. I am able to design a set of musical premises and then create an instrument to realize the music arising from them. Musical assumptions that I consciously choose dictate the design of the digital instruments I will use, rather than the instruments invoking a set of assumptions automatically. This does not deny the influence that the instruments I create have on the music I write; the point is that the initial decision about many of the preconceptions embodied in those instruments (and the working environment I create around them) is mine.

Armed with this ability, I began to approach music by considering the underlying assumptions I wanted to make in my design of instruments and the prejudices I wanted to imbed in my computer environment. I soon realized that the traditional way of working with computer instruments, tediously entering exact data at the terminal, is not a particularly pleasant way to create sound. It certainly isn't musically stimulating. Imagine again the three guitar-music situations described above. Something vital seems missing in the computer scenario. I identify this as a rich sense of interaction. In the first scenario I am intimately involved, physically, with the production of sound. I can respond immediately to what I hear. In a sense I am interacting directly with the sound itself, the interaction taking the form of my changing physical relationship with the guitar. The second scenario is striking in that my relationship to the sound produced is complicated and remarkably indirect, since a performer is involved. The performer is able to interpret because of the inexact and incomplete nature of musical notation. A discussion of the problematic nature of communication using symbolic notation systems (especially music) is far beyond the scope of this paper. Suffice it to say that when two individuals with different personal histories read the same symbol, a wide variety of responses may result.

Seen in comparison with the other two scenarios, the computer-music description appears somewhat impoverished. It has neither the direct physicality of the first scenario, nor does it have the interactive complexity of the second situation. There is indeed an interaction between the computer and myself, but it generally functions at the level of mistyping some value and producing an unexpected result. Given this unpleasant state of affairs, I decided to design an interactive environment using the computer that would be more musically satisfying. I wanted to choose a set of musical assumptions that would establish a working environment that I would find congenial.



I chose as my model a situation that I have found to be musically fruitful for me in the past. Prior to my involvement in the world of computer music, I had done much compositional work in recording studios. A typical session would consist of several people working together to compose, orchestrate, and sonically “sculpt” the music. By “sculpting” I mean the aural manipulation of sound using the signal-processing and sound-synthesis-tools available in a recording studio. Much of the art of studio recording involves the alteration of sounds through mixing, equalization (filtering), spatial processing, multitracking (arranging), and other audio-production techniques. The sounds produced in a recording-studio environment are the result of a multifaceted collaboration. Recording studios are designed to facilitate sonic experimentation; participants can suggest new ideas and experiment with possibilities quite fluidly.

I really enjoy the interaction in studio sessions. I find the interplay between my ideas and the ideas of others as music is shaped very stimulating. Much of my computer-music work had dealt with the development and use of signal-processing algorithms that are copies or extensions of the tools I utilize in the recording-studio world. I had developed in effect a virtual recording studio. All that was lacking was the easy collaboration that defined for me the recording-studio interaction. This was the difficult part of the model—getting the computer to act as a collaborator with me to make music.

## HOW DOES IT WORK?

As I reflected upon my recording-studio experiences, it struck me that the heart of the interaction I was trying to model lay in the way musical ideas were communicated. Of course, there are many features of the studio I could point to which make positive contributions to the working environment (the real-time nature of the equipment used; the ease with which signal-processing equipment is configured for various sonic effects; even aesthetic factors such as lighting, comfortable seating, room acoustics, and so on); but it is the way that musical concepts are shaped and the evolution of those concepts that is of particular interest to me. I believe that this evolution of ideas occurs through the process of communication. As thoughts are exchanged among participants in a recording project, translation (as some ideas are incompletely described) and expansion (as each individual interprets the ideas given the background of his/her own experience) of each idea communicated takes place as a matter of course.

The medium of communication used in the studio is spoken language. Typical verbal exchanges during the course of a recording session might be:

"Boost the bass on the drums."

"Try adding a little more edge to the guitar."

"Let's set up that funky bass sound we had the other day."

In each of these statements, no further information is usually necessary to create the desired action. The participants in a session more or less 'know' what is meant or implied. The brevity and ambiguity of these statements are probably the key to the fluidity of ideas in the studio.

This verbal interaction became the basic model for the Elthar program. My "spoken" requests of Elthar would be typed into the computer, with Elthar replying at the terminal screen. Several components of this model are immediately apparent.

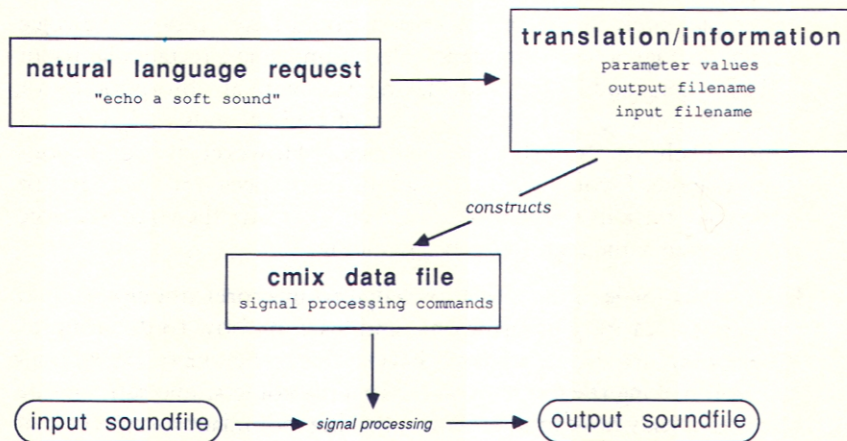
1. Some sort of natural-language interpreter is obviously needed. Not only is the syntactic and semantic freedom afforded by a natural language such as English an integral part of the interaction I wanted to emulate, the extensive descriptive capabilities inherent in such a language would also greatly facilitate musical communication. A more structured input language would eliminate much of the looseness and ambiguity I desired.
2. Elthar must be able to think for itself and use knowledge to interpret requests with varying degrees of specificity about what to do. For example, rarely in a recording session would I say "process the voice track through a room-simulation algorithm using a rectangular room 114 feet long and 77 feet wide having walls with an absorption factor of 77% and use a reverb decay time of 1.35 seconds. . . ." I would more likely say "reverb the voice track." However, it is quite conceivable that I would want to be a little more specific and say "reverb the voice track in a large room" or even "Let's try the voice in a large room with some high-frequency damping."
3. Methods for learning the knowledge to interpret under-specified requests, learning descriptions, and learning how to do complex operations are also necessary. I believe this to be one of the key factors in making the program work. In a recording-studio setting, one learns quickly how another works, making redundant specification of parameters unnecessary every time a particular sound is desired. I should stress that this is an ongoing process of learning, not merely parroting predefined sets of parameters. I wanted Elthar to be flexible enough to adapt to changing sonic conceptions and interactive enough to suggest slightly different ways to do an operation.



I also adopted the approach that Elthar should always try to generate some sort of output, even if the input request is horribly “misunderstood.” If my ideas about how musical concepts shift in this sort of interaction are correct, then it is the “misunderstandings” that will be prime contributors to the shaping of the music. I was also hoping that Elthar would be able to surprise me occasionally by doing something completely unexpected. One way to ensure this possibility is to grossly underspecify some request and have the program try its best to create some output.

#### NATURAL-LANGUAGE/KNOWLEDGE-BASE INTEGRATION

Elthar works by constructing data files in response to natural-language requests. These files are subsequently passed to signal-processing algorithms for execution. The data files contain commands to open input soundfiles for processing and open output soundfiles for writing the processed signal, along with any parameters or data needed by the requested signal-processing algorithm (see Example 1 for an illustration of this flow of command). Elthar generates all of this data by first scanning the input request for explicit (or indirect) information, and then using the knowledge base to complete any missing information.




EXAMPLE 1: ELTHAR TRANSLATES NATURAL-LANGUAGE REQUESTS INTO DATA FOR SIGNAL-PROCESSING ALGORITHMS IN THE CMIX COMPUTER-MUSIC LANGUAGE

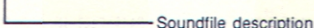
It is difficult to separate the procedures used by Elthar to parse the input sentence from the organization and operation of Elthar's knowledge base. One of the primary reasons for this difficulty is that Elthar never generates a complete parse of the sentence. Elthar repeatedly scans the sentence (or fragments of the sentence) to extract the information contained within it. The organization of these scans causes the parsing to resemble Schank's conceptual-dependency parsing methodology (Schank and Abelson 1977) coupled with Minsky's "frames" concept (Minsky 1975) for organizing program procedures and data.

The first scan done by Elthar attempts to ascertain the main verb of the sentence (I refer to this as the **ACTION** word). Elthar scans for a word by comparing each word of the sentence to words existing in a group of synonym networks, or lists of words with an identical underlying meaning or concept. The group of synonym networks being searched during a particular scan is determined by the type of scan being done. In the first scan, the group of synonym networks used is all of the **ACTIONS** known to Elthar. For example, in Example 2 the **ACTION** scan determines that the word "echoing" is a member of the synonym network for the **ACTION** **echo**. This detection terminates the scan and signals that an instance of an **echo ACTION** word has been found.


scan 1:

Try echoing a loud sound with a delay time of 0.7 and randomize the regeneration  


scan 2:

Try echoing a loud sound with a delay time of 0.7 and randomize the regeneration  


scan 3:

Try echoing a loud sound with a delay time of 0.7 and randomize the regeneration  


## EXAMPLE 2: AN EXAMPLE OF A SENTENCE-PARSING BY ELTHAR

Once the **ACTION** word has been identified, a set of procedures associated with that word are then called. This set is the **ACTION** frame. Typical procedures associated with a signal-processing **ACTION** word are more scans of the sentence to determine which input file is being requested, more scans to discover if any of the signal-processing parameters are explicitly set, and scans to extract information about any auxiliary data required by the algorithm (amplitude-envelope specifications, waveform



specifications for operations such as amplitude modulation or flanging, and so on). Each of these scans is done through the mechanism of Elthar's synonym networks, so that references to a particular specification can be quite informal.

Determining the input soundfile is important not only because the input filename is needed by the signal-processing algorithm, but also because the *filename/ACTION* word combination is used as a data-object identifier in the organization of Elthar's knowledge base. Example 3 shows the structure of a typical data object. Information about every signal-processing parameter is stored as discrete probability distributions associated with each *filename/ACTION* word data object. Every numerical value learned by Elthar for a particular signal-processing parameter has a probability of being chosen associated with it. All of the probabilities for the possible values of a signal-processing parameter form the probability distribution for the parameter. The probability distributions are discrete (rather than continuous) because Elthar's learning focuses on specific numerical values. For any parameter, however, Elthar will have a range of numerical values to choose from (the discrete probability distribution). As an example, suppose that Elthar had chosen the soundfile *thunder* as the input soundfile for the request of Example 2. After all scanning of the sentence had finished, Elthar would still be lacking a value for the *AMPLITUDE* parameter (among others) of the *echo* algorithm. Elthar would then use the probability distribution stored on the *thunder/ECHO* data object under *AMPLITUDE* to choose a value for the *AMPLITUDE* parameter. This is the mechanism Elthar uses to deal with underspecified requests. I decided to keep separate parameter-probability distributions for each *filename/ACTION* object because I believe that the use of the signal-processing algorithms depends heavily on what sound is being processed. I would use a different set of parameters for a "loud" sound than I would for a "soft" sound to achieve a certain type of effect.

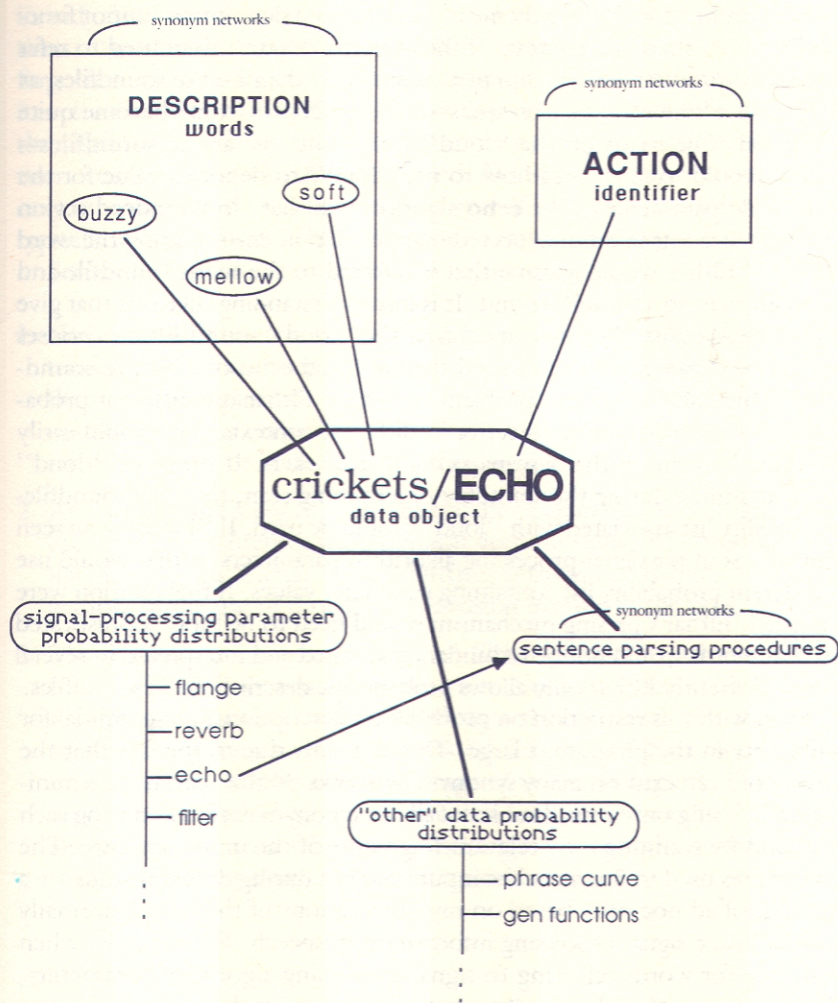
In Example 2, Elthar had to choose an input soundfile based on the description "loud." When scanning for information that can be described verbally, Elthar calls upon another set of discrete probability distributions to make choices. Each adjective has a set of probabilities associated with it denoting items described by that word. The description "loud" might be connected to the following list of soundfile names:

.31—*thunder*

.07—*rain*

.14—*wind*

.48—*crash*



EXAMPLE 3: A TYPICAL SOUNDFILE/ACTION DATA OBJECT.

When asked to process a "loud" soundfile, Elthar would use the above probabilities to choose a soundfile (how these probabilities are compiled is discussed later). This technique allows an item to be described in a variety of ways, with the relative "strength" of each descriptive attribute determined by the probability of the item being chosen for that attribute. Again, the adjectives are represented as synonym networks within Elthar, giving the program a complex ability to understand descriptions.



One problem with this scheme is that a general description cannot function in more than one context. If the adjective "loud" were used to refer to a signal-processing algorithm parameter rather than a set of soundfiles, as in "make a loud echo of a bizarre soundfile," Elthar would become quite confused. The items on the "loud" probability list are all soundfiles—Elthar would have no idea how to use "loud" to denote a value for the amplitude parameter of the **echo** algorithm. Elthar would indeed act on this sentence but in an unexpected manner. Upon encountering the word "loud," Elthar would assume that it referred to the input soundfile and proceed to echo a "loud" sound. It is misunderstandings like this that give Elthar personality. Perhaps an echo of the "loud" sound Elthar chooses would be closer to what I desired than a "loud echo of a bizarre soundfile." One solution to this problem is to have Elthar use different probability lists for a particular adjective in different contexts. This could easily be done by using Elthar's scans as context markers. If the word "loud" were identified during the soundfile-determining scan, then the soundfile-probability list associated with "loud" would be used. If "loud" were seen during a scan for signal-processing algorithm parameters, Elthar would use a different probability list containing parameter values. If this solution were adopted, Elthar's parsing mechanism would need to be more sophisticated so that words would not be redundantly scanned and interpreted in several ways. Currently Elthar only allows probabilistic descriptions of soundfiles.

Even with this restriction on probabilistic descriptions, the potential for ambiguity in the program is large. This is mainly due to the fact that the same word can exist on many synonym networks. Verbal confusion is minimized by using only a small subset of all the synonym networks during each scan and by scanning only relevant fragments of the input sentence. The procedures used to fragment the input sentence during different scans are a number of ad hoc rules based on my observations of the ways I normally communicate signal-processing information in speech. For example, when scanning for words referring to signal-processing algorithm parameters, Elthar only searches the synonym networks of words denoting the parameters of the particular signal-processing algorithm under consideration (identified during the initial **ACTION** scan). If a signal-processing parameter referent is found, Elthar then scans only the remainder of the sentence for the value of the parameter. This limiting of networks and chunking of the input sentence has the important side effect of making Elthar's parsing much more efficient.

Another aspect of Elthar's knowledge-base organization which also limits word searching concerns the management of Elthar's soundfile memory. Elthar can selectively "forget" or "recall" soundfiles upon demand. I added this feature after working with Elthar. I wanted to be able to access only a select group of soundfiles, but needed to shuttle different soundfiles



in and out of the group at will. I partitioned Elthar's memory into two sections: "working" memory and long-term memory. When a soundfile is forgotten, the soundfile data object is moved to "long-term" memory and it is removed from the list of active soundfiles. All references to it (including the adjective probability lists) are adjusted accordingly. The inverse operation occurs when a soundfile is recalled. During a sentence parse, Elthar only uses the soundfile data objects of soundfiles on the active list. This is typically a small subset of Elthar's entire soundfile memory.

Some observations about Elthar's sentence-parsing:

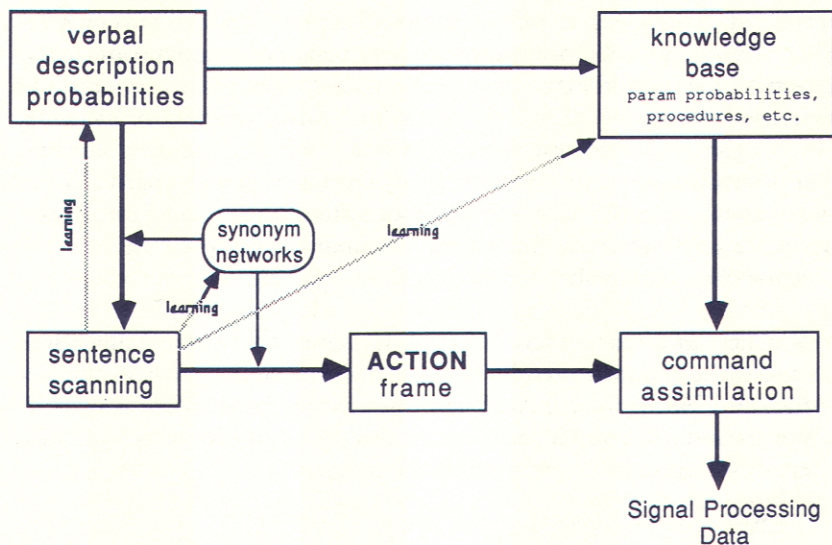
1. Nowhere in Elthar is there any explicit grammar for parsing input sentences; the grammar used by Elthar is mainly a consequence of the way the scans are organized. Although this severely limits Elthar's ability to understand many verbal locutions, it also allows an extremely wide variety of sentences to be accepted by the program.
2. As with Schank's conceptual dependency parsing methodology, radically different sentences with identical deep meanings will produce similar parsings by Elthar. The resulting actions can differ, however, because of the use of probabilistic decision making in the program.
3. The parsing/scanning system used in Elthar practically guarantees that Elthar will produce some sort of action. Even sentences such as "mix something" or simply "reverb" will cause the execution of some signal-processing operation. As I have stated, I wanted Elthar to be a very active program and to take advantage of verbal ambiguity. The exact actions taken by Elthar when presented with under-specified requests depends on the state of Elthar's working memory and on what Elthar has learned about signal processing up to that point.

## LEARNING

Elthar constructs and maintains the probability distributions it uses through three simple learning techniques: learning by being told, learning by observation, and a unique analogy-forming method for creating data structures for new soundfiles. In addition to these, Elthar can dynamically learn scripts consisting of a number of requests for carrying out complex signal-processing operations. I can also easily modify Elthar's synonym networks and other memory structures via direct interaction with the program (another case of learning by being told). Example 4 gives an overview of how Elthar's learning is integrated into the knowledge base.

The probability distributions used to make choices from adjectives are learned and altered by telling Elthar how an item is described. For instance,





EXAMPLE 4: THE GENERAL ORGANIZATION OF ELTHAR  
SHOWING WHERE LEARNING OCCURS

if I tell Elthar “the thunder soundfile is an ominous sound,” then Elthar will either enhance the probability that the *thunder* soundfile will be chosen for the *OMINOUS* description (or whatever descriptive synonym network is represented by the word “ominous”) or will construct a new description probability distribution for *OMINOUS* if none exists. This interaction can occur at any point during my conversation with Elthar. As a matter of course, Elthar asks for descriptions of new soundfiles when they are introduced in order to fully incorporate the new soundfiles into the knowledge base. Elthar also understands negative descriptions (“the thunder is not very loud”) and adjusts the adjective probabilities accordingly.

New synonyms are defined for Elthar in much the same way. I can say “the word strange means odd” or “big means large” and Elthar will add the new synonym to all of the synonym networks that have the defining word as a member. As discussed in the preceding section, I can also selectively modify Elthar’s memory of soundfiles by telling Elthar to forget or recall certain sounds. Elthar’s memories (descriptions, words, soundfiles) can be permanently altered by asking to “erase” or “delete” an item.

Elthar learns how to use the signal-processing algorithms by observing values I choose for algorithm parameters (or randomly chosen values, at my request), and soliciting my response to the operation. Elthar has a number of ‘common-sense’ constraints that are invoked when ‘randomly’ choosing

a value. For example, the program will never choose a starting point for processing a soundfile that lies beyond the end of the soundfile nor will it choose a negative duration. Many of the 'common-sense' constraints are also skewed probability distributions reflecting my experience in using signal-processing algorithms. These constraints are extremely general and designed mainly to prevent illegal values from being used—I didn't want to rule out any values that might be musically useful.

Suppose that Elthar had performed the action requested in the sentence of Example 2, and upon listening I decided that I didn't particularly care for my choice of delay time but that the value Elthar used for the regeneration seemed very appropriate. Suppose also that the attack portion of the overall amplitude curve sounded very nice. Elthar would ask me what I thought of the sound it had synthesized. Responses such as "save the regeneration value and the fade-in but not the delay time" or "I liked the fade-in and the regen factor" will cause Elthar to change the probability distributions for those parameters. This will then affect Elthar's choices for those parameters when the knowledge base is consulted in the future.

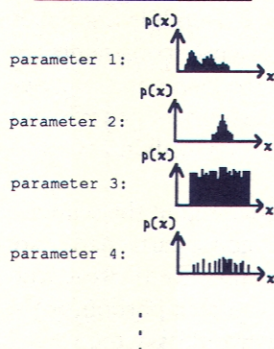
I realize that using subjective criteria (my "likes") and treating each parameter as an independent entity is quite problematic. My goal was to have Elthar track my thoughts about the sounds being processed. Elthar keeps separate probability distributions for every soundfile known. This is so subtle variations in my approach to different sounds will be reflected in the knowledge base. One of the modifications to Elthar I am planning is to allow it to introspectively modify its probability distributions. This introspection may take the form of simple numerical analysis (searching for correlations, simple linear or inverse relationships between parameters, and so on) or possibly more complex analysis of data trends and clustering such as those discussed by Michalski, et. al. (1983).

It takes some time for Elthar to develop detailed probability distributions for each soundfile. A solution to this problem is to have new soundfiles inherit data from existing soundfiles. When a new soundfile is introduced, Elthar asks what other soundfiles are like the new file. The interesting aspect of this inheritance is that Elthar creates the new soundfile data through a linear combination of the probability distributions from the analogous sounds. Thus the new file has data that is unique, but at the same time contains salient characteristics of similar files (see Example 5). This simple analogy-forming technique proved to be very useful as my work with Elthar progressed.

Elthar also has the ability to assimilate scripts into its knowledge base and recall them as commands in subsequent conversations. These scripts are simply sentences grouped together under a single command name. The sentences are interpreted sequentially as if they had been entered at the terminal. Elthar does maintain some conversational continuity throughout

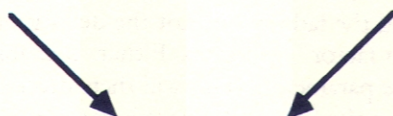
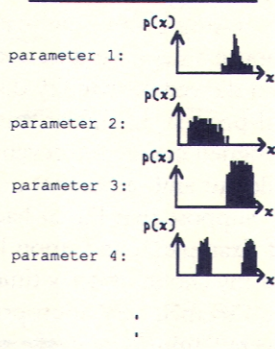


### soundfile woof

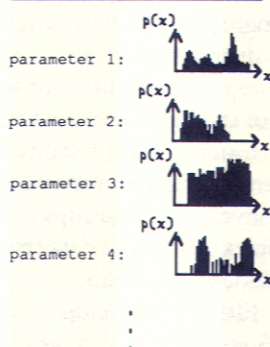


+

### soundfile meow



### soundfile mewoof



EXAMPLE 5: NEW SOUNDFILE DATA OBJECTS ARE CREATED BY COMBINING THE SIGNAL-PROCESSING PARAMETER PROBABILITY DISTRIBUTIONS OF ANALOGOUS SOUNDFILES

the duration of a script, allowing items such as the input or output file to be implied rather than explicitly referenced. Although this is more of a conversational context than is maintained during normal interaction with Elthar, it is not complete in any sense of the term. Variables or parameter values cannot be passed from one sentence to the next in the script, nor is

Elthar able to recall what operations were done previously during the execution of a script. Scripts can be recursive, but an explicit interrupt is required from outside the script to break the recursion. Even with these handicaps, complex sequences of signal-processing operations can be easily defined and used by Elthar. Example 6 shows a sample script for creating a flanged and reverberated version of an input file.

```
flange the input soundfile using a sine wave with a frequency  
of .3 Hz into output soundfile tmp
```

```
flange the input with a cycle freq of .414 into soundfile tmp
```

```
reverb tmp with a long decay time
```

#### EXAMPLE 6: A SAMPLE SCRIPT FOR FLANGING AND REVERBING AN INPUT SOUNDFILE

#### ELTHAR IN ACTION

Although the Elthar program is designed to accept virtually any English sentence related to signal-processing operations (and soundfiles) it knows about, it does expect a certain loosely defined sequence of events. This sequence is dictated by the two modes (or contexts) used by Elthar to parse sentences. The first mode is Elthar's general listening mode. In this state (signaled by the **next command?** prompt), Elthar is expecting commands for signal-processing operations or requests for information about soundfiles, algorithms, scripts, and so on. The second mode is Elthar's learning mode. It is in this mode that Elthar learns how to use the signal-processing algorithms. Some of Elthar's learning is done in the general listening mode, however. Specifically, Elthar learns soundfile classification and description during general listening. Elthar's analogy-forming ("Soundfile *X* is like soundfiles *Y* and *Z*") and description-learning ("Soundfile *X* is loud and stupid") abilities are both invoked from following the **next command?** prompt.

As discussed previously, Elthar learns by doing. Thus the learning mode is entered immediately after executing some signal-processing operation. This mode involves Elthar asking a series of questions about what was done. The answers to these questions form the basis for Elthar's learning. When the learning mode is finished, Elthar returns to general listening.

To clarify how Elthar is used, consider the record of a typical Elthar session in Example 7.



1 want me to clean up exec/?  
 2 + no  
 3 + what sounds do you know?  
 4 I have info on these soundfiles:  
 woof noise meow flamflam daytime looching howling barks babyness  
  
 5 next command?  
 6 + what sounds are described as obnoxious?  
 7 the following soundfiles are classified as obnoxious:  
 (14 (1 flamflam) (3 noise) (7 howling) (3 barks))  
  
 8 next command?  
 9 + how would you describe soundfile daytime?  
 10 I would describe soundfile daytime as:  
 mellow nice pleasant  
  
 11 next command?  
 12 + echo an obnoxious sound with a delay time of .7  
 13 what output file does this go to?  
 14 + obnokecho  
  
 15 the command file is:  
  
 open("sf/barks", 0, 0)  
 system("sfc sf/obnokecho")  
 open("sf/obnokecho", 1, 2)  
 setline(0, 0, 50, 1, 50, 1, 75, 1, 100, 0)  
 echo(0, 0, 8.072000, 1, 0.210000, 0.457000, 0.700000, 3.500000)  
  
 16 do you want to run it?  
 17 + yes  
 18 did you like that?  
 19 + yes, I did  
 20 what should happen?  
 21 + save the attack curve and the regeneration factor  
 22 Elthar understands  
  
 23 next command?  
 24 + flange something  
 25 what output file does this go to?  
 26 + flanginess  
 27 **Check the NEEDED file before running this!**

EXAMPLE 7: A SESSION WITH ELTHAR. ELTHAR'S OUTPUT IS IN **BOLD**  
 FONT, THE USER'S INPUT IS IN PLAIN FONT. LINE NUMBERS ARE  
 INCLUDED FOR REFERENCE

28 gen not found, choose one?  
 29 + yes  
 30 the command file is:  
  
 makegen(1, 10, 512, 1)  
 open("sf/babyness", 0, 0)  
 system("sfc sf/flanginess")  
 open("sf/flanginess", 1, 2)  
 setline(0, 0, 10, 1, 50, 1, 100, 0)  
 flange(0, 0, 14, 1, 0.970000, 0.740000, 0.070000, 0.500000)  
  
 31 do you want to run it?  
 32 + yes  
 33 did you like that?  
 34 + ask me later  
 35 ok, check it later  
  
 36 next command?  
 37 + what scripts do you know  
 38 You have defined these scripts:  
 craziness superflange low gotoit nocomb manycomb manyroom humpo  
 mechanization flangify interpify amify echolate  
  
 39 next command?  
 40 + make a mechanization  
 41 what output file does this go to?  
 42 + mechodaytime  
 43 any particular input file you want?  
 44 + daytime  
  
 45 about to execute:  
 comb soundfile daytime into output file tmp using freq of 100  
 and regen of .9  
 am soundfile tmp into output file tmp2 using freq of 114 and  
 a sawtooth waveform  
 flange soundfile tmp2 into output soundfile mechodaytime using  
 a sine wave and freq of 7 and regen of .87  
  
 46 the command file is:  
  
 open("sf/daytime", 0, 0)



```
system("sfc sf/tmp")
open("sf/tmp", 1, 2)
setline(50, 1, 75, 1, 100, 0)
comb(0, 0, -999, 1, 100, 0.900000, 0.077500, 0.500000)
```

```
makegen(1, 7, 512, -1, 512, 1)
open("sf/tmp", 0, 0)
system("sfc sf/tmp2")
open("sf/tmp2", 1, 2)
setline(0, 0, 50, 1, 90, 1, 100, 0)
am(0, 0, -999, 1, 114, .500000)
```

```
makegen(1, 10, 512, 1)
open("sf/tmp2", 0, 0)
system("sfc sf/mechodaytime")
open("sf/mechodaytime", 1, 2)
setline(0, 0, 10, 1, 52, 0)
flange(0, 0, -999, 1, 7, 0.870000, 0.149000, 0.500000)
```

47       processed them commands!

48       next command?

49       + stop it, no more, halt!

50       Goodbye

#### EXAMPLE 7 (CONT.)

The following describes the action of the commands and discusses Elthar's responses:

##### *Lines 1-2:*

Upon entering the program, Elthar asks if a certain subdirectory (*exec/*) should be cleaned—all files erased. The *exec/* subdirectory is where the cmix signal-processing command files are stored prior to execution. These files are not erased automatically. This is to allow them to be inspected or executed at a later time. Periodically the subdirectory must be emptied or it will rapidly become unmanageable.

Elthar also uses several other subdirectories to store vocabulary lists, probability distributions, command script definitions, etc. These subdirectories are regulated directly by Elthar and require no user intervention.

#### Lines 3-4:

Elthar has now entered general listening mode. The question causes Elthar to display a list of soundfile objects currently in working memory. This is a small subset of all the soundfiles known to Elthar—remember that the program can selectively forget and recall soundfiles (for example, “forget about the soundfile named looching” will cause *looching* to be removed from the list of active soundfiles).

If a soundfile is requested for processing and it is not in Elthar’s working memory, Elthar’s long-term memory (where forgotten soundfile-objects are stored) is searched. If the soundfile exists in long-term memory, then Elthar temporarily restores it to working memory for processing. Otherwise Elthar assumes that the soundfile is new and asks a series of questions (“what is soundfile X like?” “how would you describe soundfile X?” etc.) designed to enable the program to construct a new soundfile data object. The new object is then added to the working memory list.

#### Lines 5-7:

The **next command?** prompt signals that Elthar is again in general listening mode. The query about soundfiles described as obnoxious causes a list of the *OBNOXIOUS* soundfiles to be displayed. This list shows how most of the probabilistic attributes are represented in Elthar. The number at the beginning of the list is the total number of “units” currently in the list. The numbers associated with each entry are the number of units allotted to the entry. Thus the soundfile *howling* has a 7/14 or 50% chance of being selected if Elthar is asked to process an “obnoxious” sound. The *barks* soundfile has a 3/14 (21.43%) chance of being chosen. Elthar learns how soundfiles are described by adding soundfiles to these description lists or modifying the units associated with each soundfile.

#### Lines 8-10:

This request for information causes Elthar to display the *DESCRIPTION* words which have the soundfile *daytime* as a member of their probability list.

#### Lines 11-17:

Elthar is now asked to perform a signal-processing task. The only parameter explicitly set is the delay time (0.7 seconds). The rest of the parameters will be determined using Elthar’s learned knowledge about how to



echo obnoxious sounds. Elthar also needs to know the name of the output soundfile (the processed sound has to go somewhere!). Since the name was not given in the request, Elthar asks for it. Elthar also checks to see if this soundfile currently exists. If it does, then the output of this run will be added to it (this is how complex musical passages can be built). If not, then Elthar creates a new soundfile. Elthar fills in the unspecified parameters and then displays the cmix commands (line 15—note that the delay time is the seventh parameter of the *echo* algorithm). The user is then given a choice of executing the cmix commands now, later, or not at all.

*Lines 18–22:*

After performing the signal-processing task, Elthar switches to learning mode and asks if the user was happy with the result. The user can express total satisfaction (or dissatisfaction) or can single out particular items that Elthar did for comment. In line 21, Elthar is instructed to remember how the attack envelope and echo regeneration were done. Elthar will then increase the probabilities of the values used for those parameters.

*Lines 23–32:*

Elthar returns to general-listening mode and is given a very under-specified request. Again the output soundfile is needed, Elthar asks for this information. The message of line 27 instructs the user to check a certain file prior to execution of the cmix commands. Soundfiles are often stored off-line on magnetic tape. In this case, Elthar chose an input file (none was explicitly requested by the user) that was not on disk—the “NEEDED” file contains a list of soundfiles that must be transferred from tape before they can be processed. The *flange* algorithm also requires the specification of what is known as a *gen* function—a periodic signal used for certain effects. Since a particular *gen* function was not given by the user, Elthar asks if it should choose one. If the answer in line 29 was no, Elthar would have asked which *gen* function to use.

*Lines 33–35:*

Elthar enters learning mode after processing the soundfile, but Elthar is told to ask later about the results of the operation. Elthar will then temporarily store all of the parameters used for this particular operation so that the knowledge base may be modified later. (Currently, the later-learning program is a separate program from the main Elthar program.

It basically behaves just like Elthar in learning mode only it reads as data the temporary parameter data files constructed by Elthar when asked to “ask later.” This program could easily be incorporated into the main Elthar program, but it isn’t really necessary.) This feature was added to Elthar for several reasons. Many of the signal-processing operations take significant amounts of time to run. This feature allows many operations to be performed within a single Elthar session. It is also advantageous to allow the user to listen to Elthar’s output over a span of time before requiring that Elthar be told what was good or bad about it.

*Lines 36–38:*

Elthar is asked (and displays) a list of scripts currently defined. Elthar could also display the commands comprising any of these scripts.

*Lines 39–47:*

The “mechanization” script is requested. Elthar asks for some information missing in the script (lines 41–44) and then displays the script (line 45) and the resulting cmix commands (line 46). Unlike simple signal-processing operations, scripts are not executed from within Elthar. Instead, the cmix commands are collected into larger files within the *exec/* subdirectory for later execution. This was done mainly to ease the load upon the computer (large signal-processing jobs could be run when fewer processes were on the machine) and to allow easier inspection of the cmix data (scripts would often produce a large number of cmix commands). Elthar also does not go into learning mode after executing a script. Elthar can only learn about one algorithm at a time; a script contains many requests for different signal-processing operations.

*Lines 48–50:*

The session ends.

## WHAT DO YOU DO WITH IT?

As I worked with Elthar an interesting interaction developed that I had not foreseen when designing the program. The interaction was the result of the conjunction of Elthar’s script-learning capability with the analogy mechanism used to create data for new soundfiles. After installing the script functions into Elthar, I began to use scripts to test the program—I created several debugging scripts that called upon all of Elthar’s signal-processing commands. I purposefully left the scripts open; I didn’t specify many of



the parameters for the signal-processing algorithms. This ensured that a large portion of Elthar's knowledge base would be consulted (one of the purposes of the testing).

The sounds that resulted from these test scripts really caught my ear. They weren't at all what I had planned to do, but at the same time they had a fascinating beauty all their own. In attempting to decipher how Elthar had chosen the signal-processing parameters used to create the sounds, I began to analyze how the knowledge had been derived for the soundfiles used as input to the debugging scripts. I had originally taken time to develop fairly extensive probability distributions for several soundfiles. All of the data for other files I was using came from Elthar's analogy-forming ability. This was done mainly in the interests of time (after all, I was only debugging the program). It was much easier for me to tell Elthar what other files a new soundfile was like rather than build new data (or significantly alter analogy-created data) each time I created a new sound—this was why I developed the analogy mechanism in the first place.

My discovery of the sounds resulting from the scripts caused me to think anew about knowledge propagation through Elthar's knowledge base. The evolution of Elthar's knowledge as new soundfiles were added reminded me of the biological evolution of genetic information. I started to view soundfiles as populations, the attendant signal-processing parameter data being the gene pool surrounding the populations. The soundfiles themselves then became population phenotypes, or particular expressions of the parameter gene pools. The scripts functioned as signal-processing environments through which the soundfile populations evolved. My ears played the role of natural selection, rejecting mutant populations that didn't quite make the grade, and selecting for others whose sonic characteristics intrigued me.

This reconceptualization of Elthar changed my role in the interaction from recording-studio collaborator to experimenter in population biology. Elthar created the world in which I conducted my experiments. I began to write scripts (environments) that would select for certain characteristics—one script might foster the development of a large low-frequency content, another would tend to produce short, choppy sounds with lots of reverberation. I wondered what would happen when two populations were combined and placed in a particular environment as opposed to each one evolving independently. I repeatedly subjected certain soundfile populations to a particular script environment to see what characteristics would develop. Occasionally I would subject a population to a harsh environment designed to demolish traits that had developed over several generations. I was able to design whatever evolutionary pathways I desired.

All of these experiments left audible results (the soundfiles). The music I created with Elthar is simply a record of these tests. The evolutionary



approach I adopted produced sounds that flowed very naturally into each other. Although I did make a number of compositional choices unrelated to the experimenting while assembling the soundfiles into their final form, the basic structural ideas for the music (and the actual sounds themselves!) were a direct result of the interaction that developed between Elthar and myself. The point I want to make by relating the somewhat absurd view of myself as population scientist in sound is how much the Elthar interface influenced the music I created. Had the interaction gone differently and had I not started to think of the soundfiles as these strange populations, I would not have produced the music that I did. The interaction that developed between myself and the Elthar program affected the music at a very fundamental level.

Using Elthar as this evolution-model circumvented many of the concerns I was having about the program as I was writing it. I had thought that I would very rapidly discover the operational limits of the system. I was afraid that Elthar's operations would degenerate to a small class of sound transformations. I was concerned that these transformations would be further limited by Elthar's learning capabilities, especially given the somewhat one-dimensional character of the signal-processing parameter-learning techniques. I envisioned myself constantly pushing against the wall of what Elthar couldn't learn. This is a common complaint among users of algorithmic compositional programs (Koenig 1982; Roads 1977; Laske 1980). Some types of sounds are easily generated—those types that the process was intended to create. Other sonic conceptions are nearly impossible to realize with a given program because of the restrictions inherent in the musical assumptions made when devising the program.

I am not advocating that music interfaces should be as general as possible to allow for the creation of a wide variety of sounds from various musical conceptions. To the contrary, I believe that every interface necessarily imposes limits on what musics can be produced through it. The process of creative music-making recognizes interface limits and works within them, possibly even exploiting their restrictive characteristics. I was concerned, however, that the limits imposed by Elthar might defeat my purposes in constructing the program. Specifically, I wanted Elthar to play an active role in the sound-generation process. If the learning algorithms employed in the program caused Elthar's knowledge to converge upon a set of "lowest common denominator" values, then the transformations performed by Elthar could be easily codified. My corresponding interest in Elthar's actions would decrease as a result of this. Elthar would no longer be contributing anything to the creation process; I would be able to predict the actions of the program. Elthar would then become not much more than a souped-up signal-processing algorithm that I could use, setting certain parameters to produce expected results.



At the opposite extreme, I was worried that the complexity and randomness inherent in the program might produce wildly unpredictable output, untameable by any amount of learning. In the recording-studio model, this situation would be analogous to several people randomly setting the studio devices without listening to each other—no collaboration at all. I would certainly be able to use Elthar if this working relationship developed, but the entire program could easily be replaced by a random parameter-value generator. Elthar's role in the creation process would be no more than a noise generator in the system, creating random output for compositional filtering through me. I imagined Elthar's productions becoming enveloped in a sheen of homogeneity, each one similar to the others because of the unchanging amount of randomness generated by Elthar.

In both of these cases, I object to the amount of the compositional burden I would have to bear. The Elthar program was intended to be an active element in the sound-creation process, not merely a black box for either carrying out my specific compositional intentions or for producing a random array of sounds upon which I would pass ultimate judgement of their fitness as measured against my musical preconceptions. Through the program's learning strategies, Elthar was supposed to produce variations and suggestions relating to the musical ideas I was exploring as I explored them.

One of the paths I envisioned out of the stagnation resulting from these possibilities was to vary the source material upon which Elthar would act. Unfortunately, this does not solve the problems concerning Elthar's role in the composition process. Most of my compositional activity would take place during the selection of appropriate input sounds for Elthar. Feedback from Elthar would indeed occur in the form of my collecting new input sounds based on the transformations Elthar did to previous sounds. This interaction is much shallower than the rich sense of feedback and knowledge flow that I had hoped to capture in the program. The center of composing activities would not involve Elthar as an integral part. I would be choosing sounds to feed to some sound-transforming automaton. The two worst-case situations would represent different degrees of control over this automaton. Elthar would not be acting with much knowledge of the sounds being processed. In fact, most of Elthar's descriptive intelligence would be superfluous if the program were used in this manner.

The introduction of new sound sources was another concern I had about the Elthar program. I was afraid that the knowledge of individual sounds developed by Elthar would be so specific or idiosyncratic that it could not be applied effectively to new source material. The evolution-model approach to Elthar effectively removed this fear by making the channeling of knowledge from known soundfiles to newly constructed files one of the main features of the entire operation. This approach also eliminated



the entire set of problems surrounding the selection of new sound-source material for Elthar. New soundfiles followed quite naturally from earlier sounds because they were in fact derived from existing sounds known to Elthar. The derivation itself was even performed by the Elthar program. Any questions concerning how the new sounds might relate to Elthar's current operations and operating knowledge vanished. The operation of Elthar established sonic relationships because these relationships were fundamental to the way the program was being used.

In a similar way the evolution-model use of Elthar removed my other concerns about the program. Because this approach made the learned knowledge Elthar had accumulated essential to the sound-creation activity, my worries that Elthar's learning algorithms might be too trivial or too specific were completely unfounded. In fact, the manipulation of this learned knowledge was an explicit feature of the evolution-model mode of operation. This knowledge also constrained the possible transformations Elthar performed in a very natural way, eliminating the fear that Elthar would simply become a random value-generating machine. The knowledge Elthar used to create sonic transformations also related directly to the soundfiles through the evolutionary process. Knowledge about the sounds evolved as the soundfiles were transformed through different evolutionary pathways. At any stage of the sound-creation process, Elthar's knowledge was intrinsically part of where the sounds came from and where they were going. My fears of Elthar becoming tangential to the creative act vanished. It was in Elthar's world using Elthar's knowledge that any creation took place.

My use of Elthar as this evolution-model also undermined some personal issues that I take into the composition of any piece. An example: because digital sound synthesis gives me nearly unlimited control over the sounds I can create, I always grapple with the question of how much control I should exert. Especially when I am processing an existing sound for musical purposes, it is difficult for me to differentiate between forcing the sound into some preconception I have or letting the characteristics of the sound determine my musical concepts. I think that one of the hidden motivations behind the natural-language interface could probably be traced to this problem. I was hoping that ambiguous specifications on my part might allow me to hear some obvious musical features that I would miss otherwise.

The evolution-model circumvented this problem because I was rarely specifying individual signal-processing parameters. My control over sound production involved the design and construction of procedures through which sounds would quite naturally develop their own musical personalities—nothing was forced by direct intervention on my part. This issue of forcing sounds into some conception surfaced in a much different guise during the course of this project, however. I had developed some definite



ideas about the form the presentation of Elthar's output would take. As I considered the method I used to create Elthar's sounds and the sonic qualities of these sounds, questions began to form in my mind concerning the validity of my output-format preconceptions.

#### THERE'S NO PLACE LIKE HOME

One of my interests in composing is to project the impression of physical space: a sense that the music emanates from somewhere. In its first implementation, the Elthar program is designed to use signal-processing algorithms, especially those learned that tend to foster what has been termed the illusion of recording—the impression that a tape was actually recorded in a physical setting. Thus the musical bias of this implementation is towards the manipulation of real-world sounds in the tradition of *musique concrète*. The point of the Elthar project was not to create a transparent interface, but to work with an interface that emphasized a particular view of music and composition.

Appropriately, I chose three recordings for use as sound sources which had very strong time/place associations. The first recording is the sound of birds outside my home one spring morning; the second is a recording of an approaching summer rainstorm; and the third consists of sounds recorded during the early evening hours of an August night. *There's No Place like Home* is a sonic tour of these three recordings: I used Elthar to weave this source material into a musical context. This connective tissue, or web of relations and distinctions among the three sound sources, came about not as part of some prior plan I had for using Elthar, but as the result of my population-scientist model in which variety is produced by successive evolution from a few sound sources rather than many.

Musical associations are created by the fact that subjecting different sound sources to the same signal-processing techniques produces results that have related characteristics. For example, one particular signal-processing algorithm, the comb filter, has the ability to endow the processed sounds with definite pitches by virtue of sharp resonances created in the audio spectrum. The sounds processed using this algorithm are striking in that unlike most of the other processed sounds produced with Elthar, pitch material is present. Elthar was able to learn how to produce different "chords" and to create pitches at different points along the spectrum using this algorithm. I could say to Elthar "produce a low-pitched sound using the birds soundfile as input" and "make a low pitch from the rain recording" and Elthar would produce two files equivalent in the sense that they both contained low pitches created with the comb-filter algorithm, but derived from different sources. As with all of the parameter specifications, I



could be as specific as I wanted when instructing Elthar to use this algorithm, for example, "process the birds through the comb filter using a pitch of 149Hz." In keeping with my collaborative approach, however, I generally let Elthar choose specific parameters within learned ranges.

Similarly, I used Elthar's various room-simulation algorithms to put groups of sounds within the same acoustical space—placing two vastly different soundfiles in the acoustic environment of a simulated cave has the effect of unifying them to a certain degree: both of the sounds appear to be coexisting in the same place. The other signal-processing algorithms also imparted their particular sound to a greater or lesser extent to the processed soundfiles.

More subtle than these obvious sonic signal-processing relations is the fact that Elthar, using the analogy mechanism for creating new entries in the knowledge base, could affect different soundfiles in similar ways. I was able to establish sonic relationships between different sounds by telling Elthar that certain soundfiles were "like" other sounds. By doing this, I would essentially be telling Elthar "use the same set of presuppositions about how to generate signal-processing parameters for these soundfiles." This meant that Elthar would choose similar processing methods when asked to modify the analogized soundfiles. If I had told Elthar that the *thunder* soundfile was similar to the *booming* soundfile and then issued a series of "echo the thunder sound" and "echo the booming sound" commands, Elthar would tend to choose similar echo values for all the files produced because of the probabilistic way that parameter values are chosen. This led to the creation of entire subgroups of soundfiles that were similar to each other in that they had all been processed using related combinations of signal-processing parameter values.

Not all of Elthar's signal-processing scripts had this homogenizing effect, however. Some tended to amplify small differences, especially the scripts that recursively applied a signal-processing algorithm to a soundfile. One particular script would process an input soundfile using the flange algorithm, and then process the output of that operation through the flange algorithm, and then flange the output of that operation, eventually finishing after seven or eight iterations of this procedure. If the original sound source contained a periodic signal that aligned with the flange sweep rates, the result of this processing script would be a tremendously amplified and distorted version of that periodic signal. If the original sound source contained a periodic signal that didn't align with the flange sweep rates, then that signal would tend to smear into the flange rates. Techniques such as this are powerful tools for emphasizing or coloring subtle differences between sounds.

After several months of research, I had generated nearly two hours of



sound in the form of discrete soundfiles, ranging in length from thirty seconds to several minutes. My next task was to arrange these soundfiles into a coherent musical form. To do this, I used the Elthar program as a sonic word processor; Elthar's ability to mix and juxtapose different chunks of sound is comparable to the way paragraphs and sentences of text can be shifted with word-processing software. (The ease with which this operation can be done on the computer creates another compositional prejudice. Computer programs for audio mixing tend to encourage a view of music as a series of sound events or fragments that can be combined to create a piece.)

The mode of working that I first adopted with the Elthar interface was biased towards producing progressions of soundfiles—one group of soundfiles would be processed to produce another group, the members of which would then be processed to produce a third group, and so on. This progression from relatively simple signal-processed sounds to sounds with heavy amounts of signal-processing gave *There's No Place like Home* its linear momentum. The fact that all the sounds were derived from three original sound sources endowed the entire piece with an aural unity. The music begins with almost completely unprocessed sounds. Through the actions of the various signal-processing techniques, sonic features begin to evolve from the original sounds. These features become more and more distinct as the processing is repeatedly applied, eventually culminating in a section in which heavily processed and highly differentiated sounds are densely layered. This dense section is then gradually thinned by introducing sounds (derived from the dense sounds) that have been homogenized by other signal-processing scripts. These sounds then yield to natural sounds again with some heavily processed sounds lingering in the background, recalling the sonic features produced by the signal-processing algorithms.

The combination of this variety of soundfiles into *There's No Place like Home* created a time-dependent, highly constrained presentation of a sequence of sounds intended to give a listener a unified musical experience. The attempt to create clear musical continuity had the effect of blurring rather than highlighting some of the individual characteristics of the soundfiles produced by Elthar. I consequently decided to try two musical experiments in alternative ways to present Elthar's sounds, the goal being to project qualities of musical individuality inherent in the different soundfiles, and to give listeners a different slice of Elthar's sound world. The first experiment involved a museum or art-gallery mode of listening, the idea being that in a museum or gallery, observers are able to apprehend what is being presented in a nonlinear fashion. Objects on display in a gallery may be visited at leisure, in any particular time-ordering or sequence. Even repetitive (or circular) viewing is possible—often invited—as visitors choose their path through the gallery. To realize this idea, I set up a room with five



different displays. Four of the displays presented information (verbally and graphically) about how I constructed the Elthar sounds, the fifth display consisted of an audio mixer with instructions for operating it. At each display was a set of headphones so that observers could hear the sounds being discussed or (in the case of the fifth display) hear the sounds being mixed through the audio mixer.

Another motivation for this style of presentation was to communicate a sense of my population-researcher working method. Each of the four main displays dealt with a particular population group of Elthar's sounds. The information presented concerned the different population experiments I did to create the sounds. The fifth display gave participants a chance to produce their own version of an Elthar piece—representative sounds from various Elthar populations were played simultaneously through the different input channels of the audio mixer.

Presenting the sounds in this format produced interesting results. My "scientific" view of Elthar's music was communicated in an easy, informal manner. The "evolutionary" relationships between the sounds were made explicit through the grouping of different generations at each display (and because these relationships were discussed in the written information available at the displays). Because each soundfile was discussed and played separately at the displays, listeners could gain an appreciation for the individual sounds. By taking the sounds "out of time," I allowed the audience to form their own connections between different soundfiles and to spend as much time as desired with a particular set of sounds. Even the simple fact of hearing the music through headphones presented a different perspective. (Some of Elthar's sounds are most apparent only through headphones. An example: Elthar could process sounds using an algorithm designed to simulate the acoustics of different-sized rooms. On several occasions, Elthar chose rooms with very odd sizes [perhaps a room 250 feet long and one foot wide]. The full effect of a sound processed in this manner can only be heard through headphones.)

One of the most interesting results from this experiment had to do with the perception of where the music came from. In *There's No Place like Home*, the music is heard as sounds resulting from the processing of natural sounds. In this gallery presentation, the emphasis was more on the process of processing, rather than the results of the processing. The music arose from hearing the actions of the signal-processing techniques themselves. The difference might be stated like this: in the piece, the music was created by using signal-processing algorithms; in the gallery presentation, the music *was* the signal-processing algorithms. This seeming contradiction with my stated intention to highlight the individuality of the Elthar sounds can be explained by considering that the actions of the signal-processing algorithms can best be heard through the processing of a large variety of



sounds (as presented in the gallery room). Also, the separate presentation of each soundfile made the transformations wrought by the signal-processing techniques much clearer.

The second experiment I performed directly engaged the idea of “environments” in music. I decided to try reintroducing the Elthar sounds back into the original physical environment where they were recorded. I placed a number of speakers around the area where the three source recordings were made. I then made a series of tapes containing Elthar’s sounds—each tape was very sparse (the sounds were widely separated in time). This was done because the tapes were played back simultaneously, and I did not want a dense presentation of the music. Such a presentation would have been counter to the character of the original environment.

The overall effect was quite interesting. Freed from the responsibility of defining a sonic landscape, the sounds took on a much different character. Rather than acting as aural windows onto some other world, the sounds were actually part of the world surrounding the listener. The sounds became more vivid, almost as if they were objects (or at least caused by some existing objects) in the environment. Hearing the sounds in this fashion allowed some of their “natural” differences to become prominent. It was much easier to discern the variety of soundfiles generated by Elthar because they could exist as independent entities. By not being grouped into a piece, the sounds were able to gain a sense of individuality—even more so than with the museum/gallery experiment—due to the perceived physical independence of each sound in the environment.

I was also struck by the effect that the sounds had on the physical environment itself. Although in some sense the sounds were alien to the original environment, they had connections to it by virtue of being derived from recordings made in the original environment. Hearing these not-quite-part-of-this-world sounds tended to focus listening upon all sounds occurring in the environment, casting a different light on the normal sounds present. Even a simple “is this sound real or synthetic?” decision implies a closer inspection of the natural environmental sounds. I suspect that a large part of the vividness of Elthar’s sounds was due to this heightened listening. This aspect of the experience also had lasting effects through the renewed appreciation of the normal environmental sounds.

## RETROSPECT

The experiments done with the Elthar sounds were interesting not only for the musical experiences they presented, but also for an approach to composition they represented. In essence, by thinking about how I would present the sounds, I had extended the idea of composition to include *composing the form of the presentation*. It is obvious that the performance

environment of a piece plays a large part in the perception of the music. Most composers implicitly or explicitly take this into account when composing. Music is usually written for some (real or imagined) particular setting, most often a concert hall or some sort of performance space. However, active design of the output format of a piece gives composers the ability to shape an important aspect of the musical experience.

A major goal of the Elthar project was to explore the musical effects of a working interface. The effects were indeed dramatic; the population-scientist mode of interaction that developed with Elthar changed my entire approach to creating sounds. This fruitful Elthar interaction resulted mainly from two features of the Elthar program—the analogy mechanism and the script-defining ability. The Elthar program is complex enough that future work with the program is almost certain to define new kinds of interactions. The population-scientist mode is only one possible approach. What is particularly interesting about the program is the ability to modify it to cause variations in the interface. A simple change, such as having Elthar negate all the parameter probabilities to produce sounds unlike anything done before, would be trivial. More extended possibilities suggest themselves. Elthar could be configured to apply musical knowledge learned in one domain to another, or the program could apply signal-processing concepts to the building of harmonic materials for a piece. With a suitable connection (probably MIDI) to the outside world, Elthar could act as a real-time system, modifying the sounds produced in response to user input. The unique aspect of a real-time Elthar system (as opposed to other real-time systems) is the ability to learn about what it is doing as it performs and the ability to make future decisions based on this acquired knowledge.

This concept represents another extension of the idea of composing, an extension that appears most exciting. As I work more with music realized using computers, I have found that the choice of an interface is a major compositional decision. By being involved with the actual design of the interface, I can realize musical ideas that are intimately connected to the methods of production. Designing the interface becomes part of my compositional activity.



## NOTES

1. Octave.pitch-class notation is a way to numerically specify pitches of the western equal-tempered scale. The number to the left of the decimal point represents the octave (octave 8 is middle C), and the digits to the right represent the number of semitones above the C of that Octave. Example: 8.05 = five semitones above the C of octave 8, or the F above middle C.

## BIBLIOGRAPHY

- Koenig, Gottfried Michael. 1982. "The Aesthetic Integration of Computer-composed Scores." In *Proceedings of the International Computer Music Conference, 1982*. San Francisco, California: Computer Music Association.
- Laske, Otto. 1980. "Subscore Manipulation as a Tool for Compositional and Sonic Design." In *Proceedings of the 1980 International Computer Music Conference*. San Francisco, California: Computer Music Association.
- Michalski, Ryszard S., Jaime G. Carbonell, and Tom M. Mitchell, eds. 1983. *Machine Learning: An Artificial Intelligence Approach*. Vol. 1. Palo Alto, California: Tioga Publishing Company.
- Minsky, Marvin. 1975. "A Framework for Representing Knowledge." In *The Psychology of Computer Vision*, edited by P. H. Winston. New York: McGraw-Hill.
- Roads, Curtis. 1977. "Composing Grammars." In *Proceedings of the 1977 International Computer Music Conference*. San Francisco, California: Computer Music Association.
- Schank, Roger C., and Robert P. Abelson. 1977. *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures*. The Artificial Intelligence Series. Hillsdale, New Jersey: Lawrence Erlbaum Associates.