Elthar - A Signal Processing Expert that Learns

Brad Garton
Princeton University Music Department
Princeton, New Jersey 08540 <...!princeton!winnie!brad>

ABSTRACT: Elthar is a computer music interface designed to function with incomplete or inexact specification of parameters. Elthar's natural language interpreter translates user requests into a command format for the activation of various signal processing algorithms in the CMIX digital synthesis environment. If the user does not supply all of the required parameters for a given algorithm, Elthar consults a large probabilistic knowledge base for the needed values. This knowledge base is compiled and maintained by the Elthar program through observations of how the algorithms are employed by the user. Elthar is also capable of learning descriptive attributes and creating new data through a unique analogy mechanism. The choice of input influences Elthar's selection of signal processing parameters, allowing Elthar to draw upon different operational features for different types of sounds. This paper gives an overview of the Elthar system and comments upon the author's experience in using the program to create the piece There's no place like Home.

INTRODUCTION

Working with computers to create music during the past several years has made me recognize the integral role played by the interface between my sonic imaginings and their realization. When I pick up a pencil and paper to write a "conventional" piece of music, I automatically adopt a self-established set of assumptions that will fundamentally affect what I compose. Sitting at the keyboard of a terminal does the same thing -- different assumptions, of course. Throw in a mouse and I guarantee that the "set of pieces I might compose" will shift. Different tools make certain activities easier, certain other activities more difficult. This is bound to have at least a subtle effect on what the tools are being used to create.

"Throwing in a mouse" points to an aspect of computers that makes them rather unique as musical tools. The interface between myself and computer sound synthesis can be changed with minor hardware changes or (more significantly) by changing the software used to interpret my commands. I find this quite intriguing; being able to control the interface I use to create music gives me a very high-level "conceptual knob" that I can use to change the way I interact with my music making. I might choose to control sound production with a set of stochastic procedures, or I might use graphics software to interpret curves I draw in certain ways, or I might even write an algorithm to translate the words and letters of this paper into sound. In each case, the shape of the resulting piece (and the way I think about it as I produce it) will be different.

The Elthar program represents an explicit attempt on my part to come to terms with this ability to design an interaction for composing. Elthar is a natural language command interpreter for performing signal processing tasks. To make the interaction between myself and Elthar more interesting, I also implemented some ideas about learning and memory organization in the program. This paper discusses some of the technical aspects of Elthar and briefly describes some of the interactions that developed as I used the program to produce the piece *There's no place like Home*.

MOTIVATIONS

I have found one of the most fruitful environments for creating music to be the world of the recording studio. 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.

I decided to use this verbal interaction as the model for Elthar. Several components of the model are immediately apparent:

- Some sort of language interpreter is needed. The syntactic and semantic freedom afforded by a "natural" language like English is an integral part of the interaction I wanted to emulate, along with the extensive descriptive capabilities inherent in such a language. A more structured input language would eliminate much of the "looseness" I desired.

- 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' long and 77' wide having walls with an absorption factor of 77% and use an RT_{60} 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 "Put the voice in a large room with some high-frequency damping".

- 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 Elthar fluid. In a recording studio setting, people learn quickly how each other works, making redundant specification of parameters unnecessary each time a particular sound is desired. I should stress that this is an on-going process of learning, not merely parroting pre-defined sets of parameters. I wanted Elthar to be flexible enough to adapt to changing conceptions of how to process sounds, and interactive enough to "suggest" slightly different ways to do an operation.

Much of what I value in the recording situation centers around the interpretation of ambiguity. This was another reason for choosing a natural language interface for Elthar; I wanted to capture the ambiguity inherent in many natural language exchanges. I adopted the approach that Elthar should always try to generate some sort of output, even if the input request was horribly "misunderstood". This reflects my experience that many "misunderstandings" in the studio have led to interesting sounds and new ways of thinking about a piece.

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 for input and output file handling along with any parameters or data needed by the requested signal processing algorithm. Elthar generates all of this data by first scanning the input request for explicit (or indirect) information, and then using its knowledge-base to complete any missing information.

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 action of these scans cause the "parsing" to resemble Schank's Conceptual Dependency parsing methodology (Schank and Abelson, 1977) coupled with Minsky's "frames" concept for organizing program procedures and data (Minsky, 1975).

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 are all of the ACTIONs known to Elthar. For example, in figure 1 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.

Once the ACTION word has been identified, a set of procedures associated with that word are then called (this 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, etc.). Bear in mind that each of these scans is done through the mechanism of Elthar's synonym networks -- 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 in Elthar's knowledge-base. Information about every signal processing parameter is stored as discrete probability distributions associated with each filename/ACTION data object. As an example, suppose that Elthar had chosen the soundfile **thunder** as the input soundfile for the request of figure 1. 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 under-specified requests. I decided to keep separate parameter probability distributions for each filename/ACTION object because 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 effect.

scan 1:

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

ACTION word

scan 2:

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

Soundfile description

scan 3:

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

Signal Processing Parameters

Figure 1

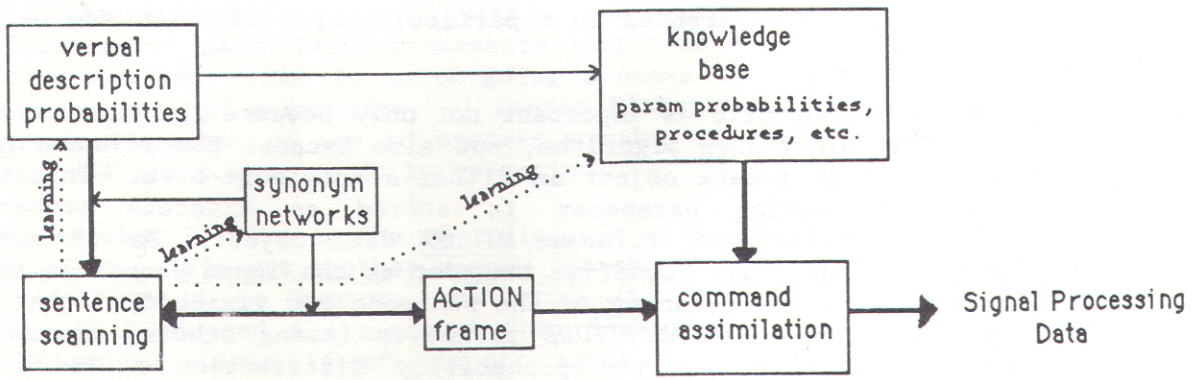

Figure 2

In the example of figure 1, 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 description word has a set of probabilities associated with it denoting items described by the word. The soundfile description "loud" might contain the following list:

(.31 - **thunder**; .07 - **rain**; .14 - **wind**; .48 - **crash**)

When asked to process a "loud" soundfile, Elthar would use the above probabilities to choose which soundfile to use. 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 description words are represented as synonym networks within Elthar, giving the program a complex description comprehension ability. The organization/interaction of Elthar's knowledge is depicted in figure 2.

There are a number of other features in the Elthar program that won't be discussed in this paper -- the techniques Elthar uses to fragment a sentence for particular scans; how Elthar generates "random" or "new" values within "common sense" limits; how Elthar handles queries about its knowledge-base, memory organization, etc. Some brief observations about the techniques I have discussed:

- 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. It was important to me to have Elthar produce output even if the input was not thoroughly understood.

- Like Schank's CD theory, 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.

- The potential for ambiguity in the program is great -- the same word may exist on many synonym networks. Verbal confusion is minimized by only using a small subset of all the synonym networks during each scan, and by scanning only relevant fragments of the input sentence. In certain cases (the description words), Elthar also checks for redundancies and issues warnings if a source of confusion is found.

LEARNING

Elthar constructs and maintains the probability distributions it uses with three simple learning techniques; learning by being told, learning by observation, and a unique analogy-forming method for creating new soundfile data structures. In addition to these, Elthar can dynamically learn scripts consisting of a number of signal processing requests. I can also easily modify Elthar's synonym networks and other memory structures through direct interaction with the program.

The probability distributions used to make choices from description words are learned and altered by my telling Elthar how an item is described. For instance, 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 and it can refer to any known item that may be understood by Elthar through verbal descriptions (usually soundfiles). Elthar will also understand negative descriptions ("The thunder is not very loud") and adjust the probabilities accordingly. New synonyms are also defined to 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.

Elthar learns how to use the signal processing algorithms by "observing" what values I choose for algorithm parameters (or "randomly" chosen values, at my request) and soliciting my response to the operation. Suppose that Elthar had performed the action requested in the sentence of figure 1. Upon listening I decided that I didn't particularly care for my choice of delay time, but the value Elthar used for the regeneration seemed very appropriate for that soundfile. Suppose that the attack portion of the overall amplitude curve was also to my liking. 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 without rote memorization. Elthar keeps separate probability distributions for every soundfile it knows, thus subtle variations in my treatment of different sounds will be reflected in the knowledge-base. One of the modifications I am planning to Elthar is to allow it to "introspectively" change its probability distributions. This introspection may take the form of simple numerical analyses (searching for correlations, simple linear or inverse relationships between parameters, etc.) or possibly more complex analyses of data trends and clustering such as those discussed by Michalski, *et. al.* (Mickalski, 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 feature 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 are unique, but at the same time contain salient characteristics of similar files. This simple analogy-forming technique proved to be very useful as my interaction with Elthar developed.

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 the duration of a script, allowing items such as the input or output file to be implied in the sentences of a script 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, nor is Elthar able to recall what operations were done previously during the execution of a script. Even with these handicaps, complex sequences of signal processing operations can be easily defined and used by Elthar.

ELTHAR IN ACTION

As I worked with Elthar to create the piece *There's no place like Home,* an 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. 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 wonder 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 were derived from these original files through Elthar's analogy-forming ability. This was done mainly in the interests of time (after all, I was only testing the program). It was much easier for me to tell Elthar what files a new soundfile was "like" rather than build new data each time I created a new sound -- after all, 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 how knowledge propagated 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 scripts functioned as signal processing "environments" through which the soundfile populations evolved. I played the role of natural selection, rejecting mutant populations that didn't quite make the grade, and "selecting for" others whose sonic characteristics interested me.

This reconceptualization of Elthar changed my role in the interaction from "recording studio collaborator" to "experimenter in population biology". Elthar became 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 was able to design whatever evolutionary pathways I desired.

All of these "experiments" left audible results (the soundfiles). *There's no place like Home* is simply a record of the empirical results 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 the piece, the basic idea for the structure (and the actual sounds themselves!) were a direct result of the interaction that developed between Elthar and myself.

CONCLUSION

The point I wanted to make by relating the somewhat absurd view of myself as "population scientist in sound" was how much the interface can influence the music that is created. Even if the influence isn't as dramatically obvious as my interaction with the Elthar program, the nature of the interface says much about the resulting art. One of the biggest criticisms of the MIDI standard is the assumptions made about music inherent in its design (Loy, 1985). The assumptions themselves may not be necessarily "bad", but it is important for composers to be aware of them and the effect they may have on creative thinking. Computers at least offer composers an alternative -- design a different interface. New MIDI software is already being developed that circumvents some of the musical decisions made by the original designers (Yavelow, 1986).

Elthar is an exploration into another type of musical interaction. It is probably a highly idiosyncratic program and may turn out to be useful for only one or two pieces. Integrating the creation of an interface into the conception of a piece seems very natural to me, however. It also allows me to take advantage of one of the unique perspectives on creativity that computers can give.

ACKNOWLEDGMENTS/INFO

Elthar is written in Franz Lisp and currently runs on a DEC Microvax II computer. The signal processing routines are written in C and run in Paul Lansky's CMIX digital synthesis environment. Elthar currently understands how to use these signal processing algorithms: flanging, amplitude modulation, reverberation (room simulation), several "flavors" of echo and delay, comb filtering, elliptical filtering (5 low-pass, 5 band-pass, 5 hi-pass), interpolation (sampling rate conversion), and mixing. I wish to thank Paul Lansky and Jim Randall of Princeton University for providing the opportunity and intellectual freedom to pursue this project.

REFERENCES

Loy, G. 1985. "Musicians Make a Standard: The MIDI Phenomenon." *Computer Music Journal* 9(4):8-26.

Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. 1983 (editors). *Machine Learning: An Artificial Intelligence Approach.* Palo Alto, California: Tioga Publishing Company.

Minsky, M. 1975. "A Framework for Representing Knowledge." *The Psychology of Computer Vision.* Winston, P. H. (editor). New York: McGraw-Hill.

Schank, R. C. and Abelson, R. P. 1977. *Scripts, Plans, Goals and Understanding.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Yavelow, C. 1986. "MIDI and the Apple Macintosh." *Computer Music Journal* 10(3):11-47.