One new application of knowledge based programming is in the area of knowledge based signal classification via the processing of acoustic data from a distributed sensor network. Here, we are investigating the automatic generation of the harmonic-operational patterns of the ocean system (Drazovich et al. 1979). In addition, we are working on basic issues such as the development of systems that can be applied in other important military mission areas, such as radar ship classification.

In the undersea surveillance application area the signal understanding program is designed to produce a description of the ocean scenario, changing with time, that encompasses all the platforms in the ocean that are communicating the signals being perceived by the sensors of the undersea surveillance system. The signals and maneuvers being detected and tracked are in a noisy ocean environment that may also cause loss of interest to the surveillance system.

Suppose an analyst is attempting to analyze signals being received by hydrophones directed toward a group of submarines or ships being maneuvered in a noisy environment. The analyst must find the location and type of each ship and associate each frequency found with a likely source. This task is known to be quite difficult. It far exceeds the capabilities of any straightforward pattern classification or pattern recognition system. It is at the limit of what is possible for knowledge based signal understanding systems consisting of large rule bases and programs that model the sources (cylinders, spheres, plates, etc.) and ratio relations of sources, types of sources on platforms, operational patterns, the ocean environment, the noise sources, maximum speeds of the platforms, whether the locations are shipping lanes, and so forth.

If the platforms decide to change their sound, they could disguise themselves effectively by changing their source characteristics, i.e., by using tone altering synthesizers, running close together, using alternate pumps and acoustic masking devices, and altering their operating patterns. The situation is made even more complicated by the introduction of new types of microphones or signal processing systems, with these kinds of changes happening all the time.

The signal understanding program would have to be reprogrammed to anticipate each possible change in data rates, harmonic structures, amplitude and frequency modulations, etc. A more reasonable approach was to anticipate all the changes and have to be able to search the very large space of possible changes to find new patterns. It is quite unlikely that it will be possible to anticipate all such changes.

A more reasonable approach might be to allow the signal understanding system to be reprogrammed to respond to new patterns. The difficulty now is reprogramming the target system in the short time allowed in a tactical situation. The necessary reprogramming and debugging is of course a slow process. A solution to the reprogramming problem is to use a program synthesis system to automatically generate and modify existing programs and data structures to meet the new requirements.

A scenario for the response to new signal characteristics might begin with the signal understanding system failing to respond, providing information inconsistent with the observed situation, or reducing certainty factors for identified sources. We assume that the appropriate action of the existing signal understanding system cannot be itself automatically synthesized, and that the associated explanation module could help pinpoint the type of signal changes that are causing the problem.

The prototype PSI system has been defined and implemented. It performs as described above. In particular, programs have been generated from English dialogues for a variety of domains. Among these programs are

- **CLASS**, a simple pattern classification program which requires only simple programming knowledge necessary for more complex programs;
- **HNS**, an information retrieval program;
- **TF**, a theory formation program which generates new patterns in an internal model of a problem; and
- **Sorting programs**, using efficient techniques for specific sorting requirements.

A new system, CH, is being developed at Systems Control for some of the potential applications discussed in the following sections.
The feasibility of many other applications may not be erratically determined. Another situation might arise from another observation the identity and location of a particular source. If so, the user could request a learning program to find patterns, expressed as rules that characterize that source. The learning program or classification system would then be automatically reprogrammed to use the new rules and reanalyze the signals. This interactive process is repeated until a satisfactory understanding of the signals is achieved.

A more difficult situation occurs when the truth of a situation cannot be established by means of any of the known criteria, rules, or meta-rules. For example, in a noisy ocean environment one can never positively identify all the submarines by the signals they produce. The analyst may suddenly notice that the number of submarines has drastically increased. One might add a constraint not anticipated by the system designer that it isn't possible for submarines to replicate themselves. The cause for the increase in sound sources might be the sound source environment was altered and the harmonics produced were taken as separate sources. It would then be appropriate to relax the constraint on groundings harmonics that previously disallowed harmonic structures would be acceptable if they arise from the same location.

Another situation might be that the sounds weren't recognized because the submarines began moving at speeds that were not anticipated. One might have the system generate its best hypothesis that anything moving very quickly or erratically couldn't be a submarine, but is an incorrect but instead a decoy. One could also add constraints for a hypothesis that best explains all possible sound sources that are least likely to miss especially interesting ones.

The first major task for our new system (called HFS) is understanding the harmonic-set formation module illustrated in Figure 1. The system we develop will input the old signal classification module, plus new signal classification rules in a language natural for expressibility producing a new automated harmonic-set formation program. This program will produce as output a modification of the original signal classification program which appropriately makes use of these new rules. The primary hypothesis is that a harmonic-set formation program will partition the set of frequency signals into a harmonically related group produced by one source of one type and the classification program will use as primitive operations (1) existing primitives of the target programming language, (2) signal retrieval commands to a data management system, and (3) subroutines in a simple statistics library.

A more difficult task is the writing of a module that learns or infers new signal classification rules on its own. The input for the module will be a list of constraints on groupings of harmonics produced were taken as separate sources. A user requests, in English, probes into the knowledge base to find patterns, expressed as rules that characterize that source. The system would then reprogram the learning program or classification system based upon prestored knowledge in a language natural for expressibility that partitions the set of frequency signals into a harmonically related group produced by one source of one type and the classification program will use as primitive operations (1) existing primitives of the target programming language, (2) signal retrieval commands to a data management system, and (3) subroutines in a simple statistics library.

The signal processing is repeated until a satisfactory change is observed in the signal processing.

The feasibility of many other applications may not be erratically determined. Another situation might arise from another observation the identity and location of a particular source. If so, the user could request a learning program to find patterns, expressed as rules that characterize that source. The learning program or classification system would then be automatically reprogrammed to use the new rules and reanalyze the signals. This interactive process is repeated until a satisfactory understanding of the signals is achieved.

A more difficult situation occurs when the truth of a situation cannot be established by means of any of the known criteria, rules, or meta-rules. For example, in a noisy ocean environment one can never positively identify all the submarines by the signals they produce. The analyst may suddenly notice that the number of submarines has drastically increased. One might add a constraint not anticipated by the system designer that it isn't possible for submarines to replicate themselves. The cause for the increase in sound sources might be the sound source environment was altered and the harmonics produced were taken as separate sources. It would then be appropriate to relax the constraint on groundings harmonics that previously disallowed harmonic structures would be acceptable if they arise from the same location.

Another situation might be that the sounds weren't recognized because the submarines began moving at speeds that were not anticipated. One might have the system generate its best hypothesis that anything moving very quickly or erratically couldn't be a submarine, but is an incorrect but instead a decoy. One could also add constraints for a hypothesis that best explains all possible sound sources that are least likely to miss especially interesting ones.

The first major task for our new system (called HFS) is understanding the harmonic-set formation module illustrated in Figure 1. The system we develop will input the old signal classification module, plus new signal classification rules in a language natural for expressibility producing a new automated harmonic-set formation program. This program will produce as output a modification of the original signal classification program which appropriately makes use of these new rules. The primary hypothesis is that a harmonic-set formation program will partition the set of frequency signals into a harmonically related group produced by one source of one type and the classification program will use as primitive operations (1) existing primitives of the target programming language, (2) signal retrieval commands to a data management system, and (3) subroutines in a simple statistics library.

A more difficult task is the writing of a module that learns or infers new signal classification rules on its own. The input for the module will be a list of constraints on groupings of harmonics produced were taken as separate sources. A user requests, in English, probes into the knowledge base to find patterns, expressed as rules that characterize that source. The system would then reprogram the learning program or classification system based upon prestored knowledge in a language natural for expressibility that partitions the set of frequency signals into a harmonically related group produced by one source of one type and the classification program will use as primitive operations (1) existing primitives of the target programming language, (2) signal retrieval commands to a data management system, and (3) subroutines in a simple statistics library.

The signal processing is repeated until a satisfactory change is observed in the signal processing.

3. Intelligence Analytic Assistant for an Automatic Ship Classification System

A knowledge-based programming system would aid an intelligence analyst by creating retrieval and classification programs that probe into the ship classification knowledge base and hypothesis structure for the current situation. Such structures are typically very large and complex, and the operational environment is constantly changing. Thus, no general-purpose ship classification system based on prestored knowledge will be capable of always presenting the correct interpretation or set of feasible interpretations. Therefore, a user such as a tactical assessment officer would be allowed to interact with the system to confirm its current hypothesis and to explore additional possibilities based upon his own past experience plus knowledge of unique properties of the current situation.

Such exploration might include requests for a summary based upon retrieval of a totally novel combination of data. For example, if many vessels aren't being identified because an important characteristic is missing or concluded, it might indicate that such vessels have been modified. A series of queries to the system via the knowledge based programming frontend might reveal important correlations, e.g., that all such vessels are of a particular type and all recently visited a particular port for an extended period.

The user should also be allowed to hypothesize certain constraints, e.g., to attempt to fill in missing data. The system would then be run to provide an analysis of the new hypothetical situation. If the system is to be an accessible assistant, the user should be able to make a complicated examination of its chain of reasoning to verify system credibility and make the system less opaque.

The only way to allow such interactions to occur, without narrowly restricting them to a priori to a small number of fixed formats, is to use a knowledge-based programming system to construct "programs which represent the required action. A human programmer who possesses the required expertise would be too expensive to assign to each ship classification system at sea, and would probably prove to be too slow at the task anyway.

4. Other Potential Applications

Another application area of interest is the automatic generation of high-level operating systems and programming systems (e.g., compilers). The target "language" might be a combination of one or more of a high-level language program, machine language program, microcode, and very large-scale integrated (VLSI) circuit board design. The particular languages produced and their combination on would depend on the specific computer configuration presented.

The feasibility of many other applications may not be far off. The promise lies in our approach, namely, that of building a large knowledge base system that emphasizes the codification of underlying programming principles combined with application-specific expertise. Some generality has already been demonstrated by extending PSI to deal with ostensibly different domains, using essentially the same knowledge base.
Figure 1: Signal Understanding Application

Surface and undersea platforms in noisy environments

- SIGNAL PROCESSING
- Frequency spectrum
- DATA MANAGEMENT PROGRAM
- RETRIEVAL PACKAGE
- HYDROPHONES
- SPOOLING DISK
- USER

Low level rules describing frequency characteristics of acoustic sources, e.g., harmonic structures, bandwidth, stability, location, etc.

High level rules describing characteristics of platforms, such as types of sources, speed, operating procedures, etc.

Sources (e.g., engines, blades, pumps)

- PROGRAM WRITTEN BY HUMAN FOR HIGHER LEVEL ANALYSIS
- HARMONIC-SET FORMATION PROGRAM (WRITTEN BY CHI)
- CHI
- Identity and location of each platform
5. The PSI Progress Synthesis System

PSI is organized as a collection of interacting modules or programmed experts, as displayed in Figure 2. A simplified view of the data paths is shown in Figure 3. There is one input data path for each specification method. Currently these are English, input-output examples, and partial traces. A more conventional method, that of a very high-level language, is a planned addition to PSI, as shown in Figure 3. These specifications are integrated into a single structure at the program net and program model levels.

PSI's operation may be conveniently factored into two parts: the input path, and the modules which are shown left of the program model, which acquires the model, and the synthesis phase, which produces a program from the model.

In the acquisition phase, sentences are first parsed, then interpreted and stored as a program net structure [Phillips-77]. The parser is a general parser which limits search by incorporating considerable knowledge of English usage. The interpreter is more specific to program synthesis, using program description knowledge, and supplies knowledge about the question asked and the current topic, to facilitate interpretation into the program net.

The dialogue moderator guides the dialogue by asking questions and giving feedback to the user. It attempts to keep PSI and the user in agreement on the current topic, provides a review and preview of topics when the topic changes, helps the user when a topic is lost, and allows initiative to the user.

The explainer module generates reasonably clear English questions about and descriptions of the program. Inference is done so very lightly that the inferred program description is the one desirable. It is also designed to explain the way and why of the acquisition and synthesis process to the interested user.

Another input specification method is a partial trace [Phillips-77]. A trace includes as a special case or partial trace all of the English. Explanations are useful for inferring data structures under simple special transformations. Partial traces of states of internal and I/O variables allow the inference of control structures. The trace and example inference expert infers a loose description of a program in the form of a program net, rather than a program model, or other true algorithms. This technique allows easy disambiguation of possible inferences and also separates the issue of efficient implementation from the inference of the user's intention.

Various types of programming knowledge are distributed throughout the modules of the acquisition phase. In contrast, knowledge specific to one particular application domain (e.g., knowledge about learning programs) is contained expert, which supplies domain support by communicating with other acquisition modules through the program net.

The program net and program model (see Figure 3) are two of the major interfaces within PSI. Both are high-level program and data structure description languages. The program model includes complete, consistent, and interpretable very high-level algorithms and information structures. The program net contains a collection of methods to acquire knowledge about a program, how and why it was acquired, and also a partial specification of the program. Since these fragments correspond rather closely to what the user has said, the program model is also interpretable by the parser/interpreter, as well as the trace and example inference module.

The program model builder [McCune-77] applies knowledge of correct program models to convert the fragments into a model. The model builder takes into account the availability of more conventional model builders, that of a very high-level language, is a planned addition to PSI, as shown in Figure 3. These specifications are integrated into a single structure at the program net and program model levels.

After the acquisition phase is complete, the synthesis phase begins. This phase may be viewed as a series of refinements of the program model into an efficient program, or, as a heuristic search in a refinement tree for an efficient program that satisfies the program model.

The coder [Barstow-75] has a body of program synthesis rules [Green & Barstow-75, Green & Barstow-77] which are applied to gradually transform the program model from abstract into more detailed constructs until it is in the target language. The algorithms and data structures are compiled independently. The coder deals primarily with the notions of set and correspondence operations and can synthesize programs involving sequences, loops, simple input and output, linked lists, arrays, and hash tables.

The refinement tree effectively forms a planning space that proposes only legal, but possibly inefficient, programs. This body is shared by the coder and the efficiency expert [Lamb-77]. When the coder proposes more than one refinement or implementation, the efficiency expert reduces the size of the tree space to a feasible size-space cost product of each proposed refinement. The better path is followed and there is no backup unless the estimate later proves to be very bad. An additional method to reduce the size of the search space is the factorization of the program into relatively independent parts so that all combinations of implementations are not considered. An analysis for bottlenecks allows the synthesis effort to concentrate on the more critical parts of the program.

6. Acknowledgements

I thank Bob Drazovitch and Roland Parks of the SIFT project at SCI for the ideas they contributed to this paper.

This paper describes research being done at Systems Control, Inc. This research is supported in part by the Defense Advanced Research Projects Agency under AFPA Order 7447, which is supported by the Office of Naval Research under ONR Task NR-349-43. The views and conclusions contained in this paper are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of SCI, AFPA, ONR, or the US Government.
7. References


[Kant-77] Elaine Kant, "The Selection of Efficient Implementations for a High-Level Language", Proceedings of the Workshop on Artificial Intelligence and Programming Languages, Stanford University, Volume 12, Number 3, SIGPLAN Newsletter, Number 5, August 1977, pages 142-145.

