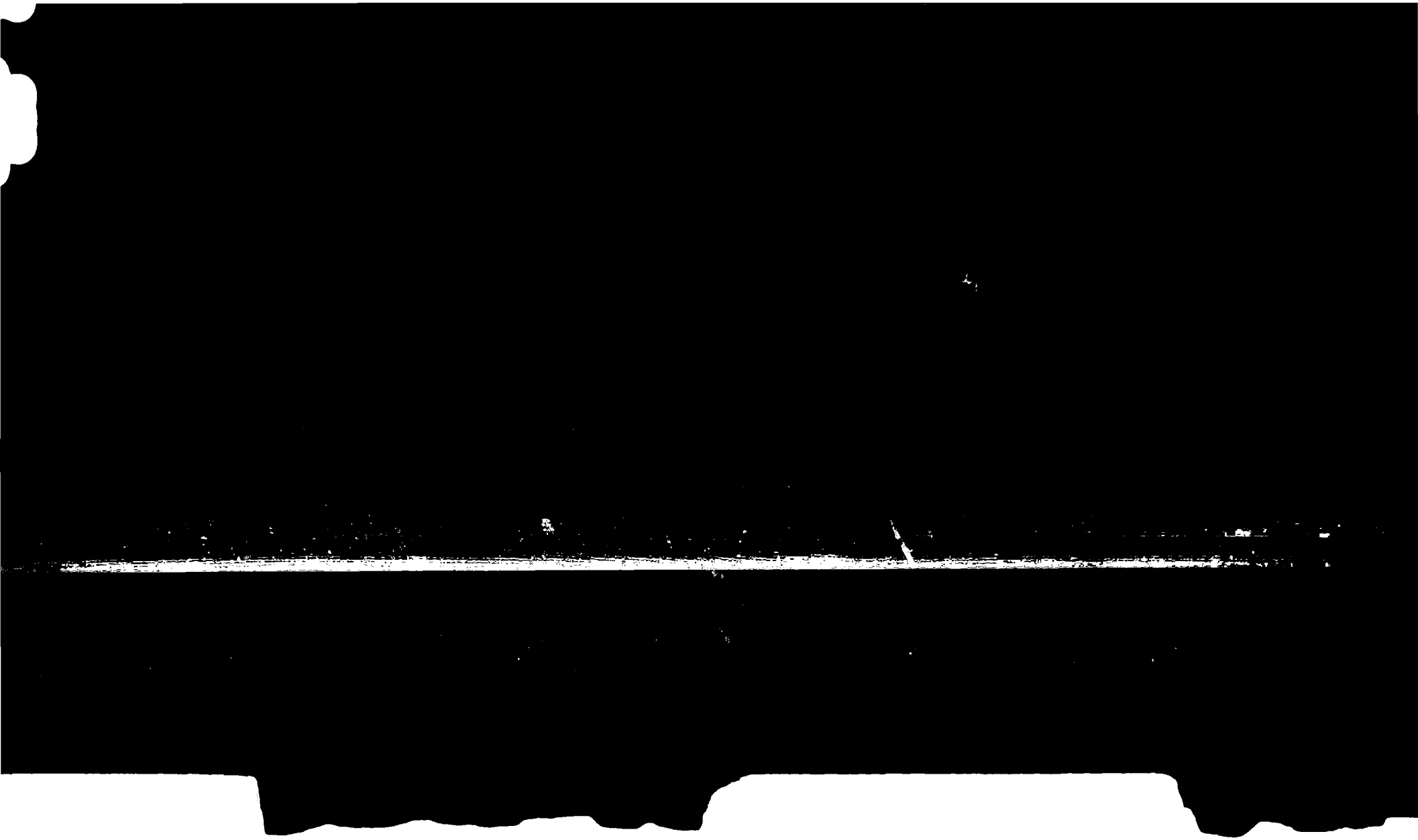
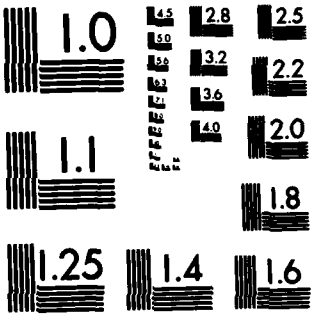
AD-A148 913 TECHNICAL WORKSHOP ON THE APPLICATION OF ARTIFICIAL INTELLIGENCE AND SPAT..(U) SYSTEMS CONTROL INC(VT) PALO ALTO CA FEB 79 N00014-79-C-0127 1/1

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Technical Workshop on the Application of Artificial Intelligence
and Spatial Processing to Radar Signals for Automatic Ship Classification

New Orleans, Louisiana
February 1979

APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION IS UNLIMITED (A)

AD-A148 913

KNOWLEDGE BASED PROGRAMMING APPLICATIONS

Cordell Green and Brian P. McCune
Systems Control, Inc.
Palo Alto, California
(415) 494-1165

February 1979

Contract N00014-79-C-0127

1. Introduction

This paper describes the knowledge based approach to program acquisition and synthesis, and is based upon an earlier work [Green & McCune-78]. Several diverse potential applications are mentioned, including synthesizing the harmonic-set formation module of an acoustic signal understanding system for knowledge based ship classification, creating retrieval and analysis programs for ship classification hypothesis structures, and making operating systems and programming languages more portable.

Progress in knowledge based systems is exemplified by a description of PSI, a knowledge based programming system that acquires high-level descriptions of programs and produces efficient implementations of them [Green-76]. Simple symbolic computation programs are specified through dialogues that include natural language, *input-output pairs, and partial traces.* The programs produced are in LISP, but experiments have shown that the system can be extended to produce code in a block structured language such as PASCAL.

A knowledge based program synthesis system works as follows. The user wants a performance program of some type, for example, a news story indexing program. The user must be versed in the application area, though need not be an expert at programming. The user conducts a dialogue, using natural language as well as traces, examples, and very high-level languages to describe the desired program. From this specification the knowledge based system synthesizes an efficient implementation of the program. Since program specification and use is an evolutionary process, each successive version of the candidate target program is tested by the user. Modifications in specification result in successive versions of the target program as new requirements develop or previous requirements are clarified.

The prototype PSI system has been designed 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 much of the programming knowledge necessary for more complex programs;
- NEWS, an information retrieval program;
- TF, a theory formation program which "learns" (forms) an internal model of a concept by repeatedly examining examples of the concept; and
- Sorting programs, using efficient techniques for specific sorting requirements.

A new system, CHI, is being developed at Systems Control for some of the potential applications discussed in the following sections.

2. Synthesis of Modules for a Knowledge Based Ship Classification System

One new application of knowledge based programming is in the area of knowledge based ship classification via the processing of acoustic data from a distributed sensor network. Specifically, we are investigating the automatic generation of the harmonic-set formation portion of the SIAP system [Drazovich et al.-79]. In addition, we are working on basic issues so that similarly developed systems 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 indicates the platforms in the ocean that are generating the signals being perceived by the sensors of the undersea surveillance system. The ships and submarines being detected and tracked are in a noisy ocean environment that may also contain other ships of no interest to the surveillance system.

Suppose an analyst is attempting to analyze signals being received by hydrophones directed toward a group of submarines and other platforms 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 parametric classification or pattern recognition system. It is at the limit of what is achievable by knowledge based signal understanding systems consisting of large rule bases and programs that model the sources (blades, shafts, pumps, etc.), harmonic 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 (e.g., by using tone altering synthesizers, running close together, using alternate pumps and acoustic masking devices, running near sound reflecting structures, and altering their operating patterns). The situation can be further complicated by the introduction of new types of microphones or signal processing systems. With these kinds of changes happening, the problem is challenging indeed. The problem is that the signal understanding program would have to be preprogrammed to anticipate each possible change in data rates, harmonic structures, amplitude and frequency modulation, etc. Similarly, any approach using learning would also have to anticipate all of the types of changes that could be expected 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 and debugging a complex 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 reprogram or 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 other observations, or reducing certainty factors for identified sources. We assume that the appropriate portion of the existing signal understanding system was itself automatically synthesized, and that the associated explainer module could describe in English the performance of the system and help pinpoint the type of signal changes that are causing the problem.

DEC 13 1984
DTIC FILE COPY
REPRODUCED AT GOVERNMENT EXPENSE

The user requests, in English, probes into the data to look for any patterns that characterize sources that seem to cause trouble. A user may know 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 signal understanding 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 controlled experiments, which is frequently the case. For example, in a noisy ocean environment one can never positively identify all the platforms to determine what is really producing the signals. Since one cannot ascertain truth, one can only judge that a given signal analysis is satisfactory according to some set of criteria. Then the program must still find new rules that produce some satisfactory model of the situation, according to criteria that an adequate model of the situation would satisfy.

The difficulty is compounded, in that it will not in general be possible to anticipate what criteria or meta-rules a satisfactory model must satisfy. For example, the analyst might 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 that some sound generator was altered and the harmonics produced were taken as separate sources. It would then be appropriate to relax the constraints on groupings of harmonics so 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 assumes that anything moving very quickly or erratically is not really a submarine, but instead a decoy. One could also add constraints for a hypothesis that best explains all possible sound sources or is least likely to miss especially interesting ones.

The first major task for our new system (called CHI) in undersea surveillance is reprogramming 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 expressing them. CHI will produce as output a modification of the original signal classification program which appropriately makes use of these new rules. The classification program is, in this case, primarily a harmonic-set formation program that partitions the set of frequency signals into a harmonically related group produced by one source of one platform. 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 hypothesizes new pattern classification rules on its own. The input for this task is a list of constraints that all rules must satisfy. The output is a program that modifies old rule sets based upon new signal information about known situations.

3. Intelligence Analyst's Assistant for an Automatic Ship Classification System

A knowledge based programming system would aid an intelligence analyst by creating retrieval and analysis 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 upon 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 should 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 occluded, 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 rerun to provide an analysis of the new hypothetical situation. If the system is to be a fully 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.

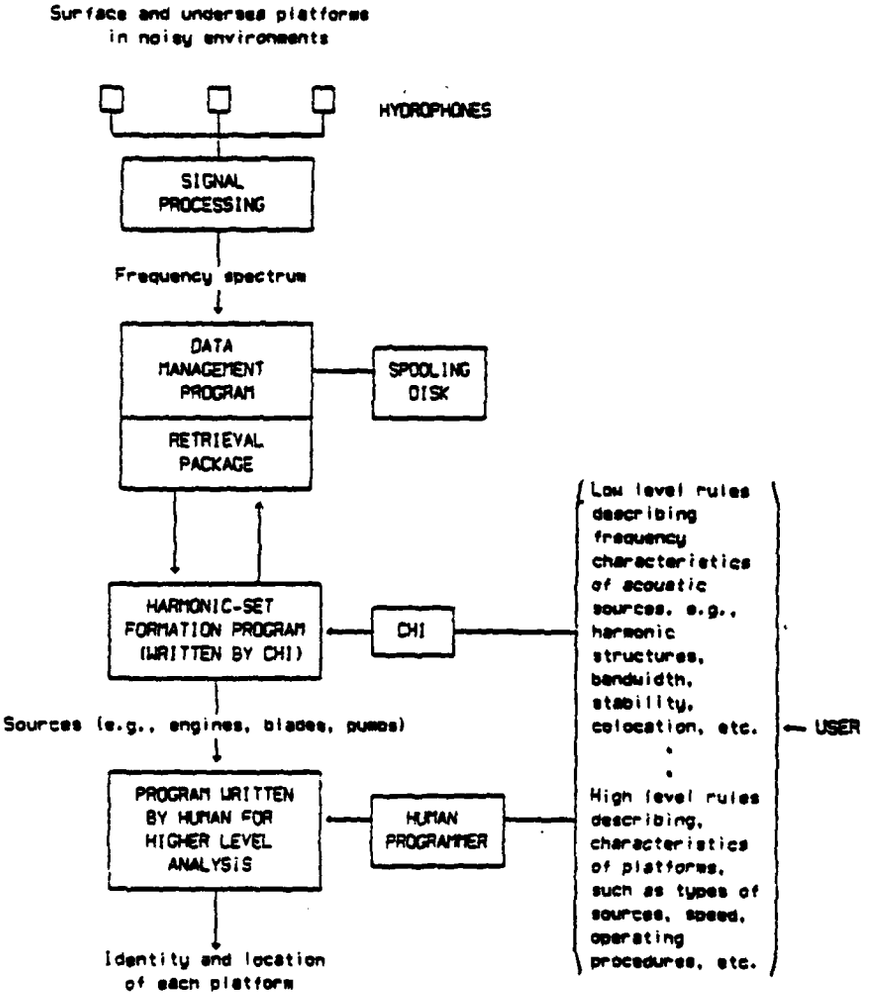
The only way to allow such interactions to occur, without narrowly restricting them a priori to a small number of fixed formats, is to use a knowledge based programming system to construct programs which represent the required action. A human programmer who possesses the required expertise would be too 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 transportable 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 would be tailored to each 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 based 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 kinds of programs, using essentially the same knowledge base.

Figure 1: Signal Understanding Application



DTIC
ELECTE
DEC 13 1984
S D
B

DTIC COPY INSPECTED

Accession For
NTIS CRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

5. The PSI Program 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 (see Figure 2): the acquisition phase (those modules 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 [Ginsparg-78]. 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, as well as knowledge about the question asked and the current topic, to facilitate interpretation into the program net.

The dialogue moderator guides the dialogue by selecting or suppressing questions for 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 who gets lost, and allows initiative to shift between PSI and the user.

The explainer module generates reasonably clear English questions about and descriptions of the program net as it is acquired, in order to help verify that the inferred program description is the one desired. It is also designed to explain the how 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 an example input-output pair. Examples are useful for inferring data structures and simple spatial transformations. Partial traces of states of internal and I/O variables allow the inductive 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 algorithm. This technique allow domain support to disambiguate 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 concentrated in the domain 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, on the other hand, forms a looser program description. Fragments of the program net can be visited in the order of occurrence in the dialogue, rather than in execution order, and allow less detailed, local, and only partial specification of the program. Since these fragments correspond rather closely to what the user says, they ease the burden of 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 processes fragments, checking for completeness and correctness, fills in detail, corrects minor inconsistencies, and adds cross-references. It also generalizes the program description, converting it into a form that allows the coder to look for good implementations. The completed program model may be interpreted by the model interpreter to check that it performs as desired by the user and also to gather information needed by the efficiency expert, such as statistics on set sizes and probabilities of the outcome of tests.

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-77] 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 algorithm and data structures are refined interdependently. 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 tree structure is shared by the coder and the efficiency expert [Kant-77]. When the coder proposes more than one refinement or implementation, the efficiency expert reduces the search by estimating the time-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. Acknowledgments

We thank Boo Brazovich and Roland Payne of the SIAP 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 ARPA Order 3687, which is monitored by the Office of Naval Research under ONR Task NR-049-433.

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, ARPA, ONR, or the US Government.

Figure 2: Block Diagram of the PSI Program Synthesis System

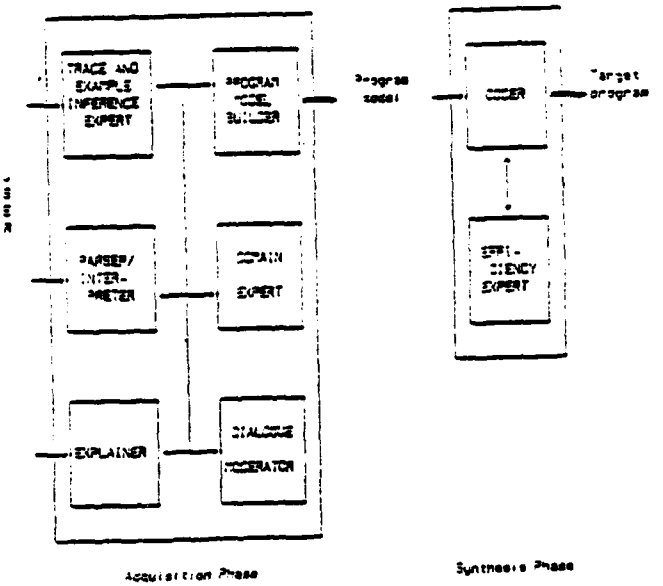
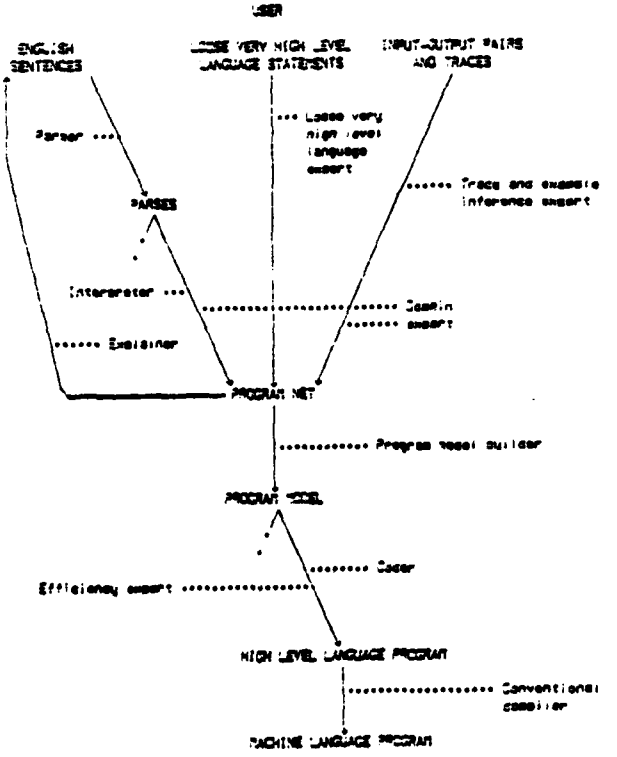


Figure 3:Major Paths of Information Flow in PSI



7. References

[Barstow-77] David R. Barstow, Automatic Construction of Algorithms and Data Structures Using a Knowledge Base of Programming Rules, Ph.D. thesis, Memo AIM-308, Report STAN-CS-77-641, Artificial Intelligence Laboratory, Computer Science Department, Stanford University, Stanford, California, November 1977.

[Drazovich et al.-79] Robert J. Drazovich, Scottie Brooks, and Scott Foster, "Knowledge Based Ship Classification", these proceedings, February 1979.

[Ginsparg-78] Jerrold M. Ginsparg, Natural Language Processing in an Automatic Programming Domain, Ph.D. thesis, Memo AIM-316, Report STAN-CS-78-671, Artificial Intelligence Laboratory, Computer Science Department, Stanford University, Stanford, California, June 1978.

[Green-76] Cordell Green, "The Design of the PSI Program Synthesis System", Proceedings Second International Conference on Software Engineering, Computer Society, Institute of Electrical and Electronics Engineers, Inc., Long Beach, California, October 1976, pages 4-18.

[Green & Barstow-75] Cordell Green and David Barstow, "Some Rules for the Automatic Synthesis of Programs", Advance Papers of the Fourth International Joint Conference on Artificial Intelligence, Volume 1, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, September 1975, pages 232-239.

[Green & Barstow-77] C. C. Green and D. R. Barstow, "A Hypothetical Dialogue Exhibiting a Knowledge Base for a Program Understanding System", in E. W. Elcock and D. Michie, editors, Machine Intelligence 8: Machine Representations of Knowledge, Ellis Horwood, Ltd., and John Wiley and Sons, Inc., New York, New York, 1977, pages 335-359.

[Green & McCune-78] Cordell Green and Brian P. McCune, "Application of Knowledge Based Programming to Signal Understanding Systems", Distributed Sensor Nets: Proceedings of a Workshop, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania, December 1978, pages 115-116.

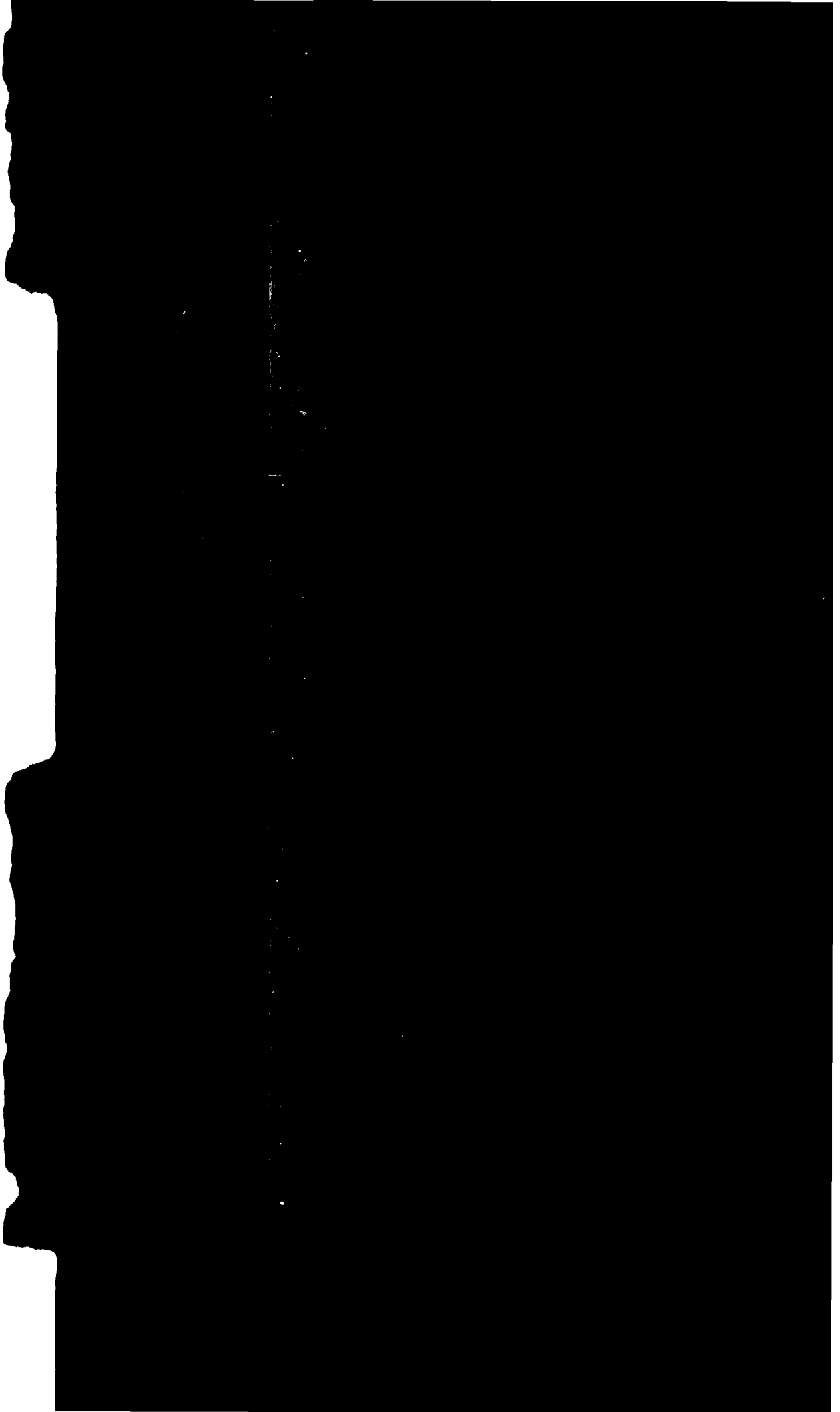
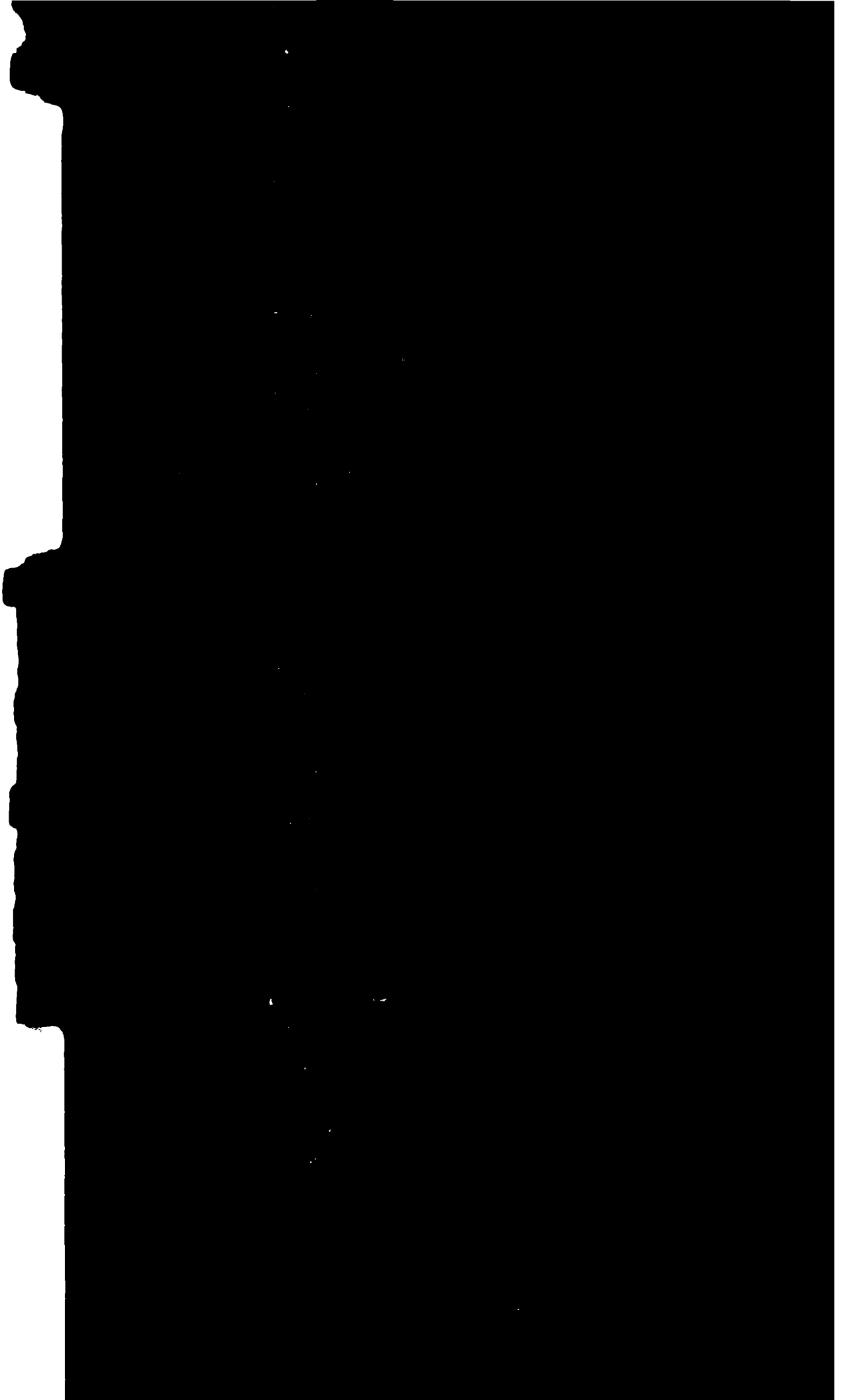
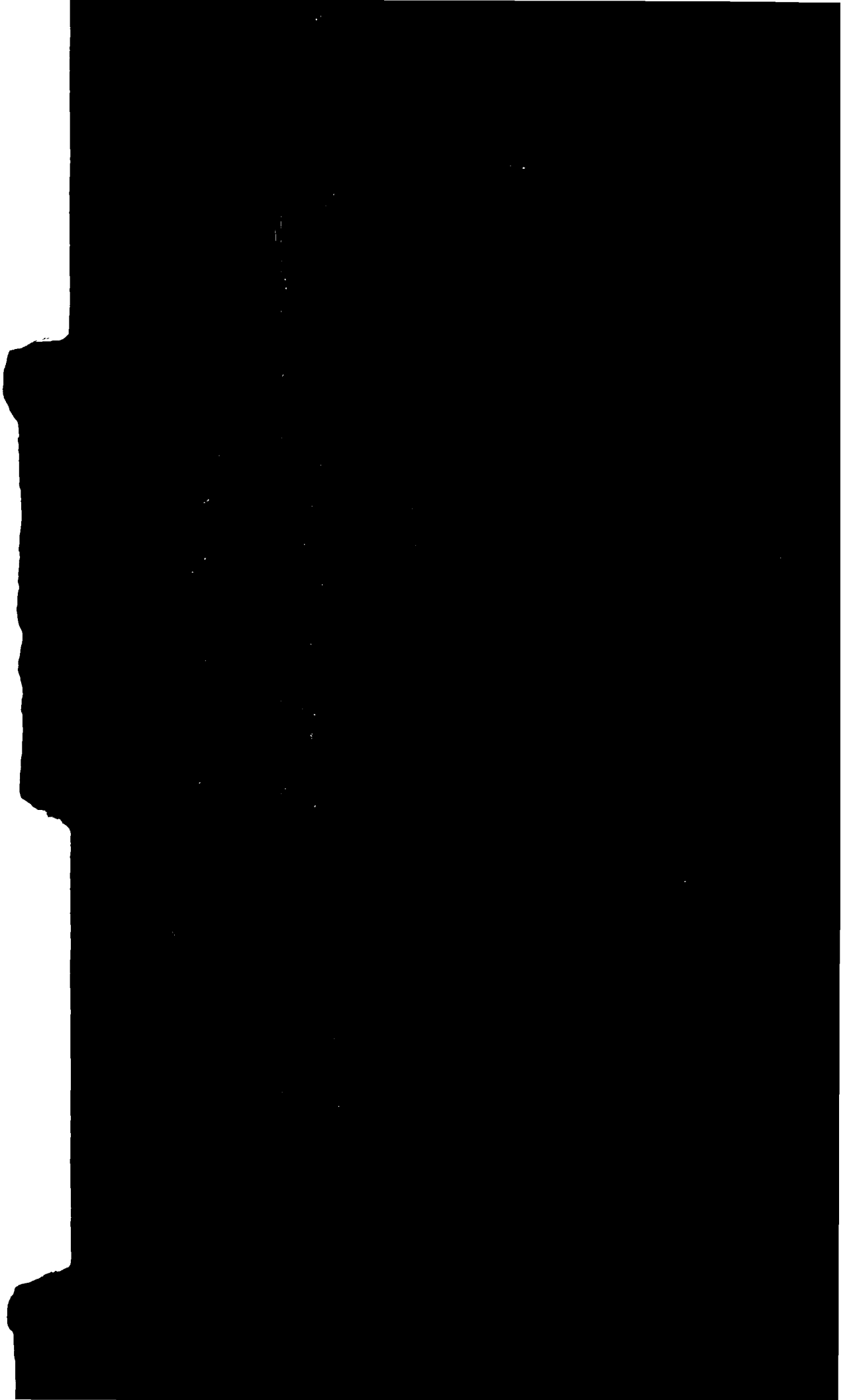
[Kant-77] Elaine Kant, "The Selection of Efficient Implementations for a High-Level Language", PROCEEDINGS OF THE SYMPOSIUM ON ARTIFICIAL INTELLIGENCE AND PROGRAMMING LANGUAGES, SIGPLAN Notices, Volume 12, Number 8, SIGART Newsletter, Number 64, August 1977, pages 140-146.

[McCune-77] Brian P. McCune, "The PSI Program Model Builder: Synthesis of Very High-Level Programs", PROCEEDINGS OF THE SYMPOSIUM ON ARTIFICIAL INTELLIGENCE AND PROGRAMMING LANGUAGES, SIGPLAN Notices, Volume 12, Number 8, SIGART Newsletter, Number 64, August 1977, page 130-139.

[Phillips-77] Jorge V. Phillips, "Program Inference from Traces Using Multiple Knowledge Sources", Proceedings of the Fifth Joint Conference on Artificial Intelligence-1977, Volume 2, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania, August 1977, page 812.

END

FILMED

1-85

DTIC