Program Inference from Traces using Multiple Knowledge Source*

Jorge V. Phillips

Artificial Intelligence Laboratory
Stanford University
Stanford, California 94S05

ABSTRACT: This paper presents an overview of a framework for the synthesis of high-level program descriptions from traces and example pairs in an automatic programming system. The framework is described in terms of a methodology and a rule base for generating control and data structure specifications for the program to be synthesized, in a format suitable for transformation into program code in a given target language.

KEYWORDS: *Automatic Programming, Program Inference, Informal Program Specifications*

One of the ways to specify a program is by the use of traces and example pairs of the program's behavior. A trace is basically a mixed sequence of operator applications, information structure states, and holdings of relations between these. Example pairs are a particular instance of traces, in which the only information present is states of program data structures. A methodology for the synthesis of programs from such descriptions must provide a means for the transformation of such descriptions into actual program code. In the past proposed methodologies have been closely tied to the coding process and have had no programming domain support [Bauer-75, Shaw-75]. In an automatic programming system, such as the PSI system being developed at Stanford [Green-761 it is highly undesirable to transform directly a program specification into code for several reasons: it is hard for the system to explain the reasoning that leads to the target program and hard for the user to verify that the system really understood the program description. It is also imperative to use some domain support to disambiguate the user's intentions. A framework has been devised for the synthesis of programs from traces and examples that copes with these constraints by using information about the target program's domain and by being totally decoupled from the coding process. TX, a program embodying this framework, has been implemented in INTER LISP, and is currently one of the constituent modules of the PSI *system.*

TX transforms input traces into partial loose descriptions (called *fragments)* which in turn are transformed by PSI's Model Builder [McCune-77] into a consistent and complete high-level program description. This description is suitable for use by PSI's Coder [Barstow-77] to produce LISP or SAIL code. The process of transformation of input traces into fragments is accomplished in several stages. The input trace components are classified and mapped into an internal trace representation using domain knowledge. This representation is a linear sequence of instantiated templates that describe interactions between the target program and its environment, transformation of data objects by the program and conditions arising in the interaction of program components. Control structure loops and tests are formed by a set of rules that detect possible iterations and associate holdings of conditions with activation of operators, using a combination of matching, unification and other inductive inference techniques. Data structures are obtained from example instances based on the syntax of the example itself and domain information, by a process of description, unification and generalization, that goes from the primitive components of the data structure to its top level description. Rules are currently available for the inference of arbitrarily complex compositions of sets, tuples, multisets correspondences and plexes, and primitive objects (strings, atoms, numbers). These partial control and data descriptions are the input to the synthesis phase of the PSI system.

Currently TX has synthesized several different classes of simple learning programs and set manipulation programs. A facility is being added to use PSI's natural language parser [Cinsparg-77] as the front end of TX to enable the user to mix freely natural language dialogues and traces. Research is being directed towards isolating a base of programming knowledge about traces and examples and attempting the synthesis of more complex and sophisticated programs. The possibility of using the structure of such a system for the acquisition of domain knowledge is under study. For further details the reader is referred to [Phillips-77]

References

Barstow-77 Barstow, David R., *A Knowledge Based System for Automatic Program Construction* to be presented at the 5th International Joint Conference on Artificial Intelligence, Cambridge, Mass. August 1977.

Bauer-75 Bauer, Michael *A Basis for the Acquisition of Procedures from Protocols* in Advance Papers of the IV International Joint Conference in Artificial Intelligence, Tbilisi, Georgia USSR, 1975

Cinsparg-77 Ginsparg, Jerrold, *A Parser for English and Its Application in an Automatic Programming System,* Ph.D. thesis, AI Memo, Artificial Intelligence Laboratory, CS Report, Computer Science Department, Stanford University, Stanford, California, forthcoming.

Green-76. Green, C. Cordell, *The PSI Program Synthesis System: An Overview* Proceedings of the Second International Software Engineering Conference, Computer Society, Institute of Electrical and Electronic Engineers, Inc. Long Beach, California, October 1976, pp 4-18

McCune-77 McCune, Brian P. *The PSI Algorithm Model Builder: A System Which Synthesizes Very High-Level Algorithms,* to be presented at the ACM SIGART-SIGPLAN Symposium on Artificial Intelligence and Programming Languages, Rochester, New York, August 1977

Phillips-77 Phillips, Jorge V. *Program Inference from Traces using Multiple Knowledge Sources* AI Memo, Artificial Intelligence Laboratory, Stanford University (in preparation)

Shaw-75 Shaw, Swartout and Green, *Inferring LISP Functions from Examples* Advance Papers of the IV International Joint Conference in Artificial Intelligence, Tbilisi, Georgia USSR 1975

Delayed Interpretation using Processing Notes

Christopher K. Riesbeck
Yale University
New Haven, CT 06511

Descriptive Keywords and Phrases

natural language understanding - semantic representation memory and inference - knowledge structures - memory models

Background

ELI (English Language Interpreter), the natural Language analyzer used by the Artificial Intelligence project at Yale (Riesbeck and Schank, 1976, Schank, 1976), builds Conceptual Dependency (CD) forms (Schank, 1975) to represent Its semantic interpretations of input texts. By including in the CD for.us special elements, called processing notes, ELI can indicate what parts of the CD form later inferencing programs must complete. Three different notes have been developed so far.

The KEF note

ELI uses the KEF note to guide the instantiation of noun phrases into memory tokens. For example, "he" is analyzed into "(PERSON CENDER MALE REF DEF)" and "soinone is analyzed into "(PERSON GENDER MALE REF IN'DEF)". "REF DEF" indicates that existing memory tokens should be looked at. "REF INDEF" indicates that a new token is needed.

The SPECIFY note

ELI uses the SPECIFY note to indicate a hole that needs to be filled in. For example, the question "Where is he?" is represented as "I want you to tell me that he is in location SPECIFY." In CD this is:

```
(CON (ACTOR HEARER <=> IITRANS TO SPEAKER
      MOBJECT (ACTOR (PERSON REF DEF)
                     IS (LOC VAL SPECIFY)))
 LEADTO (ACTOR SPEAKER TOWARD JOY INC 2))
```

The EQUIV note

ELI uses the EQUIV note to say that an object can be described with several different CDs. For example, "Mary is the one who went to Boston" is represented as "the person named Mary is EQUIV to the person who went to Boston." In CD this is:

```
(ACTOR (PERSON NAME MARY)
 EQUIV (PERSON REL (ACTOR PERSON <=> PTRANS
                    TO (CITY NAME BOSTON)))
```

A more complex EQUIV note is used for sentences of the form "Si but S2" where SI is an action from a knowledge structure (KS) such as a script or a plan (Schank and Abelson, in press). "But" in these sentences means that S2 is contrary to the intentions of the KS. The intentions of a KS are the goals associated with that KS, plus the actions that the KS says will lead to those goals.

The sentence frame "SI but S2" is represented as "S2 is not EQUIV to any event leading to an event that is a goal of the KS in SI." In CD this is:

```
( (CON S2
   EQUIV (SPECIFY I
          REL (CON SPECIFY1
               LEADTO (SPECIFY2
                       REL (KS
                            GOAL SPECIFY2)))))
  MODE NEC)
```

Conclusion

The addition of processing notes like REF, SPECIFY, and EQUIV allows not only better communication between ELI and the inference programs, but also allows us to represent the meaning of words like "but" which make meta-coimnents about the texts in which they appear.

References

Riesbeck, C. and Schank, R. C. (1976) Comprehension by Computer: Expectation-based Analysis of Sentences in Context. Yale University, Department of Computer Science Research Report 78.

Schank, R. C. (1975). Conceptual dependency: A theory of natural language understanding. Cognitive Psychology 3(4), 552-631.

Schank, R. C. (1976). Research at Yale in Natural Language Processing. Yale University, Department of Computer Science Research Report 84.

Schank, R. C. and Abelson, R. P. (in press). Knowledge Structures. Lawrence Erlbaum Press, Hi1isdale, ~N. J.

*This work was supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored under the Office of Naval Research under contract N00014-75-C-1111.