N88-16436

A LEARNING APPRENTICE FOR SOFTWARE PARTS COMPOSITION

Bradley P. Allen
Peter L. Holtzman
Inference Corporation
5300 W. Century Blvd.
Los Angeles, CA 90045

ABSTRACT

We provide an overview of the knowledge acquisition component of the Bauhaus [1], a prototype CASE workstation for the development of domain-specific automatic programming systems (D-SAPS). D-SAPS use domain knowledge in the refinement of a description of a application program into a compilable implementation [2]. Our approach to the construction of D-SAPS is to automate the process of refining a description of a program, expressed in an object-oriented domain language, into a configuration of software parts that implement the behavior of the domain objects.

We view this process of software parts composition as a problem-solving task. By structuring a problem-solving task so that the types of knowledge required are made explicit, the acquisition of knowledge useful in performing the task can be made simpler, and the resulting knowledge base becomes easier to maintain [5]. The Bauhaus incorporates a problem-solving architecture based on the RIME [7] and SOAR [3] systems that provides such a structure. In this architecture, the task of refining an initial program description is represented as a goal. A goal determines a problem space and an initial state in that space. A problem space is a set of operators that are useful in the satisfaction of a given goal. An operator transforms a state in the problem space into a new state, or creates a subgoal, or recognizes when a given goal is satisfied. The goal of refining the initial program description is satisfied when the system has composed a set of software parts to form an implementation of the program.

Operators are applied by the system by iterating through three stages:

1. Propose the set of operators that can be applied to the current state;
2. Choose an operator from the set to apply to the current state; and
3. Apply the operator, generating the next current state.

User intervention in the choice of an operator is requested when the system reaches an impasse: when no operator applies, then the system is unable to express a preference for an operator, or when the system's preferences are inconsistent. The types of user intervention that can occur correspond to the types of knowledge needed by the system to avoid similar impasses in the future. The system generalizes from observed instances of user intervention to create new operators and preferences. In this manner, the programming knowledge of the system is automatically

increased through its use as a software development tool by experienced application developers. This form of knowledge acquisition through the observation of user intervention in the design process allows us to characterize the Bauhaus as a learning system[4], similar to the VEXED VLSI design system [6]. Implementation of the Bauhaus is currently underway using ART running on a Symbolics Lisp machine under the Genera 7.1 environment, integrated with the Symbolics Ada programming environment.

References

1. Allen, B.P. and Holtzman, P.L. Simplifying the Construction of Domain-Specific Automatic Programming Systems: The NASA Automated Software Development Workstation Project. Proceedings of the Space Operations Automation and Robotics Conference, NASA/U.S. Air Force, August, 1987.

2. Barstow, D. "Domain-Specific Automatic Programming". IEEE Transactions on Software Engineering 11, 11 (November 1985).

3. Laird, J.E., Newell, A. And Rosenbloom, P.S. "SOAR: A Architecture for General Intelligence". Artificial Intelligence 33, 1 (1987).

4. Mitchell, T.M., Mahadevan, S., and Steinberg, L.I. LEAP: A leaning Apprentice for VLSI Design. Proceedings of the Ninth International Joint Conference on Artificial Intelligence, August, 1985.

5. Soloway, E., Bachant, J. and Jensen, K. Assessing the Maintainablity of XCON-in-RIME: Coping with the Problems of a VERY Large Rule Base. Proceedings of the National Conference on Artificial Intelligence, AAAI, July, 1987.

6. Steinberg, L.I. Design as Refinement Plus Constraint Propagation: The VEXED Experience. Proceedings of the National Conference on Artificial Intelligence, AAAI,July, 1987.

7. Van de Brug, A., Bachant, J. and McDermott, J. "The Taming of R1". IEEE Expert 1,3(Fall 1986).