"Deductive Plan Formation in Higher-Order Logic"

J. S. Darlington

A resolution-based theorem prover has been shown to be able to generate answers to questions concerning data expressed in first-order predicate logic. Darlington has extended this procedure by employing a form of "higher-order" logic, resulting in more efficient construction of an answer. The basis for the higher-order logic is a more sophisticated matching procedure when attempting unification of two strings, one or both of the two containing some functional expression. More precisely, each argument of the functional expression is allowed to match any subexpression of the other string, and a lambda-abstraction is performed whenever a match succeeds. The procedure is demonstrated on the classic "blind-hand" problem, but the real advantage is seen when applied to "program generation" or "plan formation" in which a loop is involved. An appropriate loop axiom is introduced, based on a formulation of the loop using mathematical induction, and the resolution-based theorem prover is then used to solve the problem.

The resulting program generation of an algorithm to compute $2^n$ is impressively efficient, and indicates the advantage of using higher-order logic. However, the future of such an approach surely must be questioned, especially in the area of automatic program generation. The process is still rather time-consuming, and its class of solvable problems is rather limited. As Darlington admits, several areas need considerable improvement. But hopes of sufficient improvement of unification strategies, for instance, seem dim compared with promises of some newer alternative approaches.

Paul R. Hudak

"Proving Compiler Correctness in a Mechanized Logic"

R. Milner and R. Weyhrauch

In this paper the first application of LCF (a logic for computable functions based on Dana Scott’s Logic) to proving compiler correctness for simple ALGOL-like language is described and a (partly mechanized) proof of correctness of a compiling algorithm for arithmetic expressions is presented.

The idea for formulation of compiler correctness adopted here was first
BOOK REVIEWS

proposed by McCarthy who formulated the problem in the first-order quantificational calculus. The algebraic approach has been championed by Burstall, Landin, and Lockwood Morris. While the use of the algebraic method seems to reduce the problem powerfully, the main aim of the paper is to show LCF to advantage on a nontrivial example. In the view of the second reviewer they succeed; but the first reviewer is not so convinced. The second reviewer believes that simplifications and substitutions seem more motivated and easier to find by being bound in lambda expressions and used by substitution at the proper time. The first reviewer observes the size of the expressions printed out in an example given in the appendix of the paper for a very simple example and wonders if a more intentional approach might not succeed better.

AKINORI YONEZAWA AND ROGER HALE

"Some New Directions in Robot Problem Solving"

R. FIKES, P. HART, AND N. NILSSON

This article is a discussion of some heuristic problem-solving techniques used in the Stanford robot problem-solving system STRIPS, with some speculation on how they could be generalized, and comments concerning the direction of AI robotics research in general.

STRIPS operates in a manner similar to that of Newell, Shaw and Simon’s GPS; it is given a goal statement to achieve, then it deduces a set of differences between the current state of its world and the goal state, and attempts to reduce these differences by choosing to perform actions from a set of permitted operations. The system constructs a plan by considering the prerequisites of each operator, and has the capability to generalize a plan to use in more than one specific instance.

In STRIPS, as well as in most other recent robot systems, the robot lives in a world constrained by some drastic simplifying assumptions: there is only one robot, which is the sole agent of change in the world, and the results of the robot’s actions are completely determinate. Extensions to the STRIPS planning algorithm are discussed which attempt to deal with environments in which not all of these conditions hold. Multiple goals can be handled with a system of numerical priorities, with constraints receiving negative values. Possible outcomes of independently occurring processes or robot actions can be assigned probabilities.