7 Kay, M. The MIND System. In *Natural Language Processing*, Austin, R. (Ed.). Algorithmics Press, New York, 1973, 155-188.

8 Schubert, L.K. Extending the expressive power of semantic networks. TR 74-18, Department of Computer Science, University of Alberta, 1974.

9 Schubert, L.K. Extending the expressive power of semantic networks. *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence*, 1975, 158-164.

10 Schubert, L.K. Extending the expressive power of semantic networks. *Artificial Intelligence 7*, 1976, 163-198.

11 Shapiro, S.C. The MIND system: a data structure for semantic information processing. R-837-PR, The Rand Corporation, Santa Monica, 1971.

12 Shapiro S.C. A net structure for semantic information storage, deduction and retrieval. *Proceedings Second International Joint Conference on Artificial Intelligence*, The British Computer Society, London, 1971, 512-523.

13 Shapiro, S.C. An introduction to SNePS (semantic net processing system). Technical Report No. 31, Computer Science Department, Indiana University, Bloomington, Revised December, 1976.

14 Shapiro, S.C. and Bechtel, R.J. Non-standard connectives and quantifiers for question-answering systems. In progress.

15 Woods, W.A. What's in a link: foundations for semantic networks. In *Representation and Understanding*, Bobrow, D.G. and Collins, A. (Eds.). Academic Press, New York, 1975, 35-82.

Semantic Network Representations in Rule-Based Inference Systems

Richard O. Duda, Peter E. Hart, Nils J. Nilsson, and Georgia L. Sutherland
Stanford Research Institute Menlo Park, CA 94025

Rule-based inference systems allow judgmental knowledge about a specific problem domain to be represented as a collection of discrete rules. Each rule states that if certain premises are known, then certain conclusions can be inferred. An important design issue concerns the representational form for the premises and conclusions of the rules. We describe a rule-based system that uses a partitioned semantic network representation for the premises and conclusions.

Knowledge-Directed Inference in BELIEVER

N. S. Sridharan and C. F. Schmidt
Department of Computer Science
Rutgers University New Brunswick, NJ 08903

The BELIEVER theory is an attempt to specify an information processing system that constructs intentional interpretations of an observed sequence of human actions. A frame-based system, AIMDS, is used to define three domains: the physical world; the plan domain, where interpretations are constructed using plan structures composed from plan units; and the psychological description of the actor. The system achieves a shift of representation from propositions about physical events to statements about beliefs and intentions of the actor by hypothesizing and attributing a plan structure to the actor.

A paradigm for approaching a part of the interpretation problem is described in this report. Understanding is viewed as a process of assimilating incoming patterns with existing knowledge and expectations. The essential process of "expectation matching" is attended to in detail and a simple example is presented to illustrate the paradigm and its possible extensions.

A Knowledge Base Organization for Rules About Programming

David Barstow[1]
Stanford University Stanford, CA 94305

Abstract

PECOS is a knowledge-based system for automatic program synthesis. Programs are specified as abstract algorithms in a high-level language for symbolic computation. Through the successive application of programming rules, the specification is gradually refined into a concrete implementation in the target language. The existence of several rules for the same task permits the construction of a variety of distinct programs from a single initial specification. Internally, program descriptions are represented as collections of nodes, each labeled with a programming concept and with other properties related to that concept. The refinement process is guided by the selection and application of rules about programming. These rules are stated as condition-action pairs, but the identification of certain rule types permits the use of various techniques for efficient rule retrieval and testing, including the determination of retrieval patterns and the automatic separation of the condition into an applicability pattern and a binding pattern.

Introduction

PECOS is a knowledge-based system that constructs concrete implementations of abstract algorithms [1]. For current experiments the specification language centers around notions from symbolic programming, including information structures such as collections or correspondences, and operations such as testing whether an item is in a collection or computing the inverse of a correspondence. Programs are synthesized by gradually refining the original specification into a program in the target language. Currently the target language is LISP (in particular, a subset of INTERLISP [10]), but experimentation with SAIL (an ALGOL-like language) is underway [8]. From a given specification, PECOS is able to construct several different implementations, differing both in representations for data structures and in algorithms for abstract operations.

PECOS's abilities are derived from a large knowledge base of rules about programming. These rules have been carefully designed and constructed to deal explicitly with various aspects of the programming process, including intermediate-level constructs and certain design decisions. In previous experiments, such rules have been used to synthesize several simple sorting programs [5,6]. Detailed discussions of all of PECOS's rules may be found elsewhere [1]. The current discussion focuses on the organization of the knowledge base and the techniques used to retrieve and apply its rules.

Rules about Programming

The rules in PECOS's knowledge base constitute an explication of knowledge about writing programs in the domain of symbolic computation. While many of the rules are relatively specific to the task of writing simple symbolic programs, some are generally applicable to programming in other domains as well. Most are independent of any particular programming language, although some are quite specific to LISP. A representative sample is given below. (The rules are presented in English for ease of understanding; details of the internal representation are discussed later.)

1. This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract MDA 903-76-C-0206. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Stanford University, ARPA, or the U. S. Government.