HPRL:A Language For Building Expert Systems

Steven Rosenberg

Computer Research Laboratory
Hewlett-Packard
1501 Page Mill Road
Palo Alto, Ca. 94304

Abstract

This paper describes an extension of FRL (Frame Representation Language) which supports the encoding of reasoning knowledge within a frame-based formalism. The extension is called HPRL (Heuristic Programming and Representation Language). The declarative representation of reasoning knowledge in the same formalism that is used to represent domain knowledge results in a powerful tool for the construction of expert systems. Reasoning knowledge is easy to describe, examine and modify. Rules can be reflexive, allowing the construction of powerful meta-rules. HPRL runs on a Vax 11/780, and on the HP-9836. It has been used for various exploratory projects at Hewlett-Packard, including a program to diagnose faults during IC manufacturing, a program for analyzing dual-channel ECG information to diagnose arrhythmias, and a program for analyzing spectra from infrared and mass spectrometers.

1. INTRODUCTION

The Applications Technology Department of the Computer Research Laboratory of Hewlett-Packard has been pursuing the development of "heuristic programming languages". As part of this effort, we have built an integrated language for the representation of knowledge and for reasoning about that knowledge. The language uses a single decarative formalism for representing domain and reasoning knowledge, including knowledge about the reasoning process itself (rule interpreters, agendas, decision trees, etc.).

Different representation languages emphasize different features. In HPRL we have tried to build a language that is a good tool for building expert systems. To do so, we embodied such design criteria as a uniform declarative representation, a particular syntax for grouping related information in frames, and so on. Other languages have emphasized different features. KL-ONE (Brachman, 1977), for example, emphasizes the epistomological significance of the particular features which its representation tools provide the user. HPRL, by contrast, emphasizes the tools themselves. HPRL does not claim to embody the "right" theory of how to represent knowledge. Instead, HPRL is presented as a tool within which the user can construct his own model.

Aside from this difference in emphasis, KL-ONE and HPRL contain a similar set of representation tools. The initial representation components were both developed at approximately the same time - 1977. Currently KL-ONE is being extended to allow for the expression of logical relations among elements in a separate syntax (Brachman and Levesque, 1982). HPRL has gone the route described in this paper of integrating reasoning and representation within a declarative semantics.

There *are* at least two other languages with similar goals to HPRL: RLL (Greiner and Lenat, 1980) and AGE (Nii and Aiello, 1979). RLL is a frame-based representation-language language whose goal is to allow the user to modify the basic representation capabilities. HPRL does not provide this degree of freedom. However, HPRL allows reasoning knowledge to be expressed in the representation language, and provides methods for extending and modifying . the reasoning capabilities. AGE is a language whose goal is to provide tools that allow a user to emulate different problem solving architectures such as EMYCIN, or HEARSAY-III. In this respect it contains a well developed set of tools. However, HPRL allows a user to design and build his own tools, as well as use existing ones.

MRS (Gensereth, Greiner and Smith, 1981) is a knowledge representation system designed to provide a single language for stating facts, while storing those facts in a variety of different representations. Unlike MRS, HPRL does not give the user control over the primitive datatypes and processes that underlie the language. Thus a user stores information in frames, but does not know how frames are implemented. He does not have the option, as in MRS, of defining how classes of information should actually be implemented (i.e. in arrays, lists, etc.).

HPRL can be compared to older reasoning languages like EMYCIN (VanMelle, 1980). HPRL is at least as powerful as these languages, since it contains the ability to do forward and backward chaining, as well as use meta-rules. In addition, HPRL separates out the notion of reasoning contexts from the semantics. EMYCIN makes contexts do double duty, as a separate representation component does not exist. As a result the ability to represent knowledge is compromised.

2. THE LANGUAGE

HPRL is an extension of FRL (Frame Representation Language (Goldstein and Roberts, 1977)), which provides the basis for the representation tools in HPRL. FRL, in turn, is based on Minsky's (1975) notion of frames. In various incarnations it is still in use as a representation language (Winston, 1980; Engelman, 1980). FRL provides a hierarchically organized, frames-based semantics with inheritance, procedural attachment and slot descriptions. The frame representation in HPRL (adopted from FRL) allows the user to organize domain knowledge according to user-selected semantic and ontological relations.

HPRL extends FRL by using frames to represent reasoning as well as domain knowledge. A core set of Lisp functions provide functionality by executing rules according to the directions in a frame-based rule interpreter. This results in a uniform knowledge representation for data, rules, the rule interpreter and the interpretation process. The first three are represented directly as frames. The interpretation process, which involves the manipulation of agendas, the creation of portions of various decision trees, the recording of sufficient information to enable backtracking, etc. is done primarily (but not entirely, for reasons of efficiency) through frames.

The use of a declarative representation means that reasoning knowledge is explicit, and can be examined and modified by other reasoning knowledge. This makes the reasoning knowledge more easily understood and built than if a procedural format were employed. Reasoning knowledge can be augmented with additional facts. For instance, a rule can have, besides a condition and action, an arbitrary amount of additional information, such as caveats and suggested uses.

Rules are represented as frames in HPRL. To be useful, a rule must be interpreted in an environment called a Rule Domain (which is itself a frame). This rule domain indicates how to interpret the rule frame. HPRL provides an initial set of tools for the interpretation of rules. These include a capacity for forward chaining, backward chaining, meta-rules, and meta-interpretation.

More powerful reasoning strategies can be constructed through the use of meta-rules. One typical way in which meta-rules are used in an expert system for the diagnosis of IC wafer flaws is the following: heuristic rules exist which examine what is known at any given time, and when possible generate hypotheses. A meta-rule exists which, whenever it notices that a hypothesis is applicable, changes the agenda to cause evaluation of that hypothesis first. Such a meta-rule allows the user to do best-first search in cases where he has heuristics that can generate good hypotheses. This use of meta-rules allows local optimization of the reasoning whenever sufficient evidence is accumulated.

Meta-rules in HPRL consist of a condition composed of a logical conjunction of (a) domain knowledge and (b) reasoning knowledge. The most important reasoning knowledge is that which reflects on the reasoning process itself, such as (i) when a goal gets placed on the agenda, (ii) when a goal succeeds, and (iii) when a goal fails. Other reasoning knowledge can be used, such as which rules are being considered, or are available, and whether a particular rule has succeeded. A small set of primitives allow meta-rules to alter the agenda. A meta-rule can examine the interpreter itself, and change its composition. For instance, given the failure of a goal representing a given strategy, a meta-rule might change the component of the rule interpreter that applies rules, going from one best-first search method to another.

Meta-reasoning can occur by having reasoning about the current rule domain go on in a separate rule domain. For instance, given evidence for a bad component of an IC wafer, it is possible to use the knowledge of the structural relations and manufacturing processes embodied in the domain knowledge to decide what goals to pursue next. Such reasoning occurs in a separate meta-domain. The rule interpreter can be set to always choose its next goal based on the outcome in the meta-domain. In fact, this would be very wasteful, since only some cases benefit from this kind of reasoning. Instead, we use meta-rules to notice when such situations exist. These meta-rules then directly invoke the meta-domain.

To use rules which are represented declaratively, it is necessary to provide a procedural invocation. In HPRL, this is done by creating a Rule Domain. A rule domain contains a set of rules (although rules can be part of more than one domain), and a set of instructions for interpreting the rules. The default case for the rule set is the entire rule database. The rule domain is itself represented as a frame.

HPRL provides two pre-defined domains backward chaining, and forward chaining. The user can construct others, either from scratch, or as sub-domains of these two. If a new domain is constructed which is subordinate to one of the existing domains, it will inherit the rule interpretation of its parent domain, if new instructions are not specified. The rules used would be those specified in this new domain, together with any rules specified in the parent domain. Rule domains which are subordinate to the basic domains have the effect of partitioning the rule set. This means that rule domains can be used to partition problems into sub-problems and associated rules. This is useful when sub-problems can be identified and a restricted set of rules is known to be sufficient. Within each domain, the rules will be highly relevant to the problem. This avoids excessive search through a large set of rules. It also facilitates the structuring of problems in terms of psychologically meaningful contexts. In an expert system under development at Hewlett-Packard which attempts to diagnose dual-channel ECG's, an initial categorization of the beat is used to partition the solution process into fifteen separate domains. In this case all domains had identical rule interpreters. The division into separate domains preserved a conceptual partitioning, and allowed small sets of rules to be applied in relevent cases.

Alternatively, rule domains can be used to represent distinctions in control. A user can customize a domain by examining the rule interpretation instructions, and then writing or choosing a special function for the component which defines the appropriate feature of the interpretation process. Modifying a rule interpreter is not a trivial task. However, it is made easier through the use of a declarative format for the rule interpretation features, inheritance of shared features, which localizes and minimizes the changes required, and the use of "canned" or predefined components and tools for modifying features.

3. WHAT CURRENTLY EXISTS

HPRL runs in PSL (Griss, Benson and Maguire, 1982) on the VAX and the HP-9836, a 68000-based machine. However, all new development work is occuring in PSL on the 9836, and the other versions are becoming obsolete. The PSL version has been recoded for speed and efficiency. It takes advantage of the PSL capability to write machine level code to optimize frequently performed operations.

4. APPLICATIONS

HPRL has been developed in the context of multiple application domains. These include expert systems for fault diagnosis in IC manufacturing, for the analysis of ECG arrhythmias, and for the analysis of spectra from infrared and mass spectrometers. HPRL is also being used as part of a natural language processing system. None of these applications is "complete", although some of them demonstrate substantial ability. Each exercises different aspects of HPRL.

We plan to continue to test and develop HPRL in an environment of multiple applications. The applications drive the development by showing us shortcomings in the current version. At the same time, multiple applications ensure that HPRL is not optimized for just one type of task. Multiple application domains are more likely to result in good representation and reasoning principles.

5. CONCLUSIONS

Although several expert systems are in various stages of development in HPRL, it is still too early to tell if this approach will provide significant advantages in very difficult tasks. We do not yet know if this declarative approach to interpretation will prove to be inefficient or clumsy for implementing some desirable control architectures. In addition, there are quest.ons of size and efficiency when HPRL is used to which we cannot yet answer. However, initial results with the current applications are encouraging enough that we plan to continue development of HPRL.

ACKNOWLEDGMENTS

I would like to thank Douglas Lanam, who has been responsible for the current implementation, and the members of our application groups (Reed Letsinger, Jan Aikins, Enrique Ruspini, Chris Clare, and Elaine Heron).

REFERENCES

Brachman, R. J. "A Structural Paradigm for Representing Knowledge," Ph.D. dissertation. Harvard University, Cambridge, Mass. 1977.

Brachman, R. J. and Levesque, Hector J. "Competence in Knowedge Representation," proc. Second National Conference on Artificial intelligence, Carnegie-Mellon University, Pittsburgh, Pa. 1982.

En gel man, Carl, Scarl, Ethan A., and Berg, Charles H. "Interactive Frame I nstantiateion," Proc. First National Conference on Artificial Intelligence, Stanford University, Palo Alto, Ca. 1980, pp. 184-186.

Cenesereth, Michael R, Creiner, R. and Smith, D. E. "MRS Manual," Stanford University Heuristic Programming Project Memo HPP-81-6, December 1981.

Goldstein, I. P. and Roberts, R. B. "NUDGE, A Knowledge-Based Scheduling Program," Proc. of IJCAI-S, 1977, pp. 257-263

Greiner, Russel, and Lenat, Douglas B. "A Representation Language Language," Proc. First National Conference on Artificial Intelligence, Stanford University, Palo Alto, Ca. 1980, pp. 165-169.

Griss, M. L., Benson, E., and Maguire, G. Q. "PSL: A Portable LISP System," Proc. ACM Symposium on LISP and Functional Programming, Pittsburgh, Pa. 1982.

Minsky, M. "A Framework for Representing Knowledge," in P. H. Winston (Ed.) The Psychology of Computer Vision, McGraw-Hill, N.Y. 1975.

Nii, H. P. and Aiello, N. "AGE: A Knowledge-based program for building knowledge-based programs," Proc. of IJCAI-6, 1979, pp. 645-655.

VanMelle, W. "A Domain-Independent Production-rule System that Aids in Constructing Know/edge- based Consultation Programs," Ph.D. Dissertation, Heuristic Programming Project, Dept. of Computer Science, Stanford University, Memo HPP-80-22, June 1980.

Winston, Patrick H. "Learning and Reasoning by Analogy/" Communications of the ACM, 23, 12 (Dor. 1980]. nn 689-703