The PRL Mathematics Environment: A Knowledge Based Medium

Extended Abstract*

Joseph Bates

TR 86-768
July 1986

Department of Computer Science
Cornell University
Ithaca, NY 14853

* Submitted to Workshop on High Level Tools for Knowledge Based Systems, Columbus, Ohio, October 7-8, 1986.

The PRL Mathematics Environment: A Knowledge Based Medium

Extended Abstract[1]

Joseph Bates

Department of Computer Science
Cornell University
Ithaca, New York 14853

July 1986

[1]Submitted to Workshop on High Level Tools for Knowledge Based Systems, Columbus, Ohio, October 7-8, 1986.

The PRL Mathematics Environment: A Knowledge Based Medium

Extended Abstract*

Joseph Bates
Department of Computer Science
Cornell University
Ithaca, New York 14853

July 1986

1 Introduction

PRL is the generic name for several mathematics environments designed and built over the last decade at the Cornell Computer Science Department [1,2]. PRL is a mathematical medium—it provides facilities whereby mathematicians can gradually accumulate and use large amounts of knowledge. Much of this knowledge looks roughly as one would find in mathematics books: definitions, theorems, and examples expressed directly in appropriate notation, but in addition, the knowledge is active. We claim that PRL is a prototype for a significant new class of AI systems.

The content of PRL libraries is active in two ways. First, there is domain knowledge, the material easily identified in books, that often implicitly describes means of carrying out useful computations. For example, a discussion of context-

*Submitted to Workshop on High Level Tools for Knowledge Based Systems, Columbus, Ohio, October 7-8, 1986.

free languages may convey how to compute parse trees from sentences. Once such a discussion of CFLs is in PRL, the system can extract and execute the implicitly defined parser generator. Second, libraries include meta-knowledge that describes how to assist mathematicians in extending the knowledge base. Typically libraries contain user developed heuristics for verifying "obvious" claims, for carrying out new styles of proof, and for extending the editors and other system facilities in useful directions—all of these requirements changing as the mathematics is developed in one direction or another [3].

PRL is not a knowledge based system in the conventional sense. There is no particular collection of knowledge extracted in advance from experts and built into the system. Instead, PRL provides a setting for mathematicians to gradually express the content of domains that interest them. The system can and has been taken in many directions. So PRL's declarative knowledge base is not fixed.

Further, because users build tools for a domain as they express other domain knowledge, PRL isn't even a knowledge aquisition system—with tools provided by the system builders and knowledge provided by users. Rather, as we shall discuss, it is an integrated environment in which building and using take many forms within a continuous spectrum of activities. We call the kind of system that supports such a spectrum a *knowledge based medium.*

It is our belief that knowledge based media are an interesting, powerful, but little explored class of AI systems, that may serve in many domains besides mathematics as a means for accumulating large amounts of information that will mean something to machines. They provide a setting in which to study the range of "knowledge activities", including representation, reasoning, acquisition, discovery, and learning. In addition, they provide a means for getting those people with domain knowledge directly involved in the process of building knowledge based systems—not just filling in rules or frames, but thinking about the whole enterprise and having means to work along the entire spectrum discussed above.

With this view, our natural goal is to abstract the mathematics out of PRL and work toward a more general knowledge based medium. After examining the specific mechanisms that serve well in PRL, we will describe our efforts toward that goal.

2 The PRL Mathematics Environment

PRL is a fancy editor. It is not much like an algebraic manipulator (for example, Macsyma) or an automatic theorem prover, though it has those things as parts. People write things (build things) in PRL, save their work, later read it or use it in larger works. The objects they build are saved in a library, which can be viewed at a high level or expanded so the content of individual objects can be seen.

Objects in the library are of three varieties: definitions, theorems, and meta-language expressions. Definitions describe new linguistic forms. Theorems state and prove propositions. Meta-language expressions extend the system in several different dimensions.

Most information is entered into PRL using a structure editor. Definitions are the means by which the editor's templates are created. They have the form *left-side* == *right-side* where *left-side* is a sequence of characters with interspersed parameter slots and *right-side* is an expression built with existing templates that may include instances of the parameters. For example,

$$let\ \langle x\rangle, \langle y\rangle = \langle p\rangle\ in\ \langle t\rangle == ((lambda(\langle x\rangle\langle y\rangle)\langle t\rangle)\ (car\langle p\rangle)\ (cdr\langle p\rangle))$$

$$let\ \langle w\rangle, \langle x\rangle, \langle y\rangle, \langle z\rangle = \langle p\rangle, \langle q\rangle\ in\ \langle t\rangle == let\ \langle w\rangle, \langle x\rangle = \langle p\rangle\ in\ let\ \langle y\rangle, \langle z\rangle = \langle q\rangle\ in\ \langle t\rangle.$$

Since the new notations are never parsed, only displayed, there are no parsing problems. The user is free to choose arbitrary forms with only the usual stylistic concerns about ambiguity.

Definitions create notation, but it is incorrect to think that they are "only" notation. People's and machine's ability to think depends on the forms in which they can express thoughts. Definitions describe notations, which play a role in determining potentially related concepts, but of more significance they focus attention on the meaningfully changeable parts of expressions. By providing a means of abstracting and chunking larger structures, definitions play a central role in reasoning and explanation in PRL.

Theorems state and prove propositions. They are verified by the system and are built with its help. PRL uses a constructive type theory as its mathematical foundation, so the propositions are part of an uncommonly rich language—for example, it is not possible in general to tell whether a phrase is a well-formed proposition because, as in natural languages, well-formedness is as complicated as truth. This rich computational language and its proof system provide the representational power needed to state and prove the claims arising in mathematics. Note that this is not a matter of having a universal language of computation—a proof system is different from a programming language.

Here are two example of theorems that have been proved in PRL libraries.

Theorem: Find Roots

$\forall I:\text{CompactIntervals},\ \forall f:\hat{I} \rightarrow R,$

if f is continuous on I and $f(\text{left end of } I) < 0$

and $f(\text{right end of } I) > 0$ then $\forall \varepsilon : R^{+},\ \exists x : \hat{I}$ s.t. $|f(x)| < \varepsilon$

Theorem: Pumping Lemma

for any DFA, M, there is an integer n s.t. for any word, z,

if $z \epsilon \mathcal{L}(M)$ and $|z| \geq n$ then $\exists u, v, w : \text{words}$ s.t. $z = u \cdot v \cdot w$

and $|u \cdot v| \leq n$ and $|v| > 0$ and for any integer, k, $(u \cdot v^{k} \cdot w) \epsilon \mathcal{L}(M)$

In both examples all of the notation used is from definitions, that is, none of it is built into the PRL base language.

Meta-language expressions are code written in the ML language of Edinburgh LCF [4]. The terms, goals, and proofs of PRL are represented as abstract types in ML. A major use of ML is to write tactics: functions of goals or proofs that return (partial) proofs or fail. Tactics may be combined using tacticals, for example *Repeat* f, f *Then* g, f *OrElse* g, to conveniently express simple backtracking searches. More sophisticated heuristics search the library for relevant lemmas, simplify formulas, and reason about equalities.

Tactics can be invoked from the proof editor as proof methods, in which case the partial proof they build is hidden and any remaining gaps in the proof are presented to the user for further development. They may also be invoked as transformation methods, in which case they are applied to a fragment of a proof and the new proof they produce replaces the fragment.

This simple (much improvable) mechanism for extending the proof rules and the editing operations has been critical for having PRL remain a useful system as libraries of domain knowledge grow. Together with the rich mathematical foundation and the definition facility, this mechanism has allowed users of PRL to take the system in the directions that interest them. Libraries have been built for parts of number theory, finite automata theory, real analysis, type theory, denotational semantics, and programming with arrays. Tactics take care of filling in many steps that the mathematician wants to gloss over. We take it as a sign of success that the system designers can no longer give suitable demonstrations—one has to know the libraries to understand PRL in its varied manifestations.

3 Knowledge Based Media

A KBM is an environment in which knowledge grows. The medium can be used at different levels. We see new PRL users begin by reading a library; then they try modifying a theorem or definition. Once they get the idea of PRL as a medium, they want to build some domain of their own. The first attempt is awful and they come to understand the need for meta-knowledge. They re-think the structure of their domain and then build carefully, extending the system with new tools (tactics) as is appropriate. People who have done this several times see the value in investigating AI technology and in making substantive extensions to the knowledge tools in the KBM. Some of them eventually want to re-design the system. We have seen PRL take non-AI'ers, even anti-AI'ers, and draw them into the knowledge engineering enterprise.

Mathematics and Computer Science professors consider it their task to study, organize, and explain their fields. They write books to record their knowledge and convey it to others. The ultimate goal of PRL is to have these people-who-explain working in a KBM, instead of with paper, gradually expressing their knowledge (meta-knowledge, tools, etc.) in an active, machine understandable form.

In thinking about high level tools for knowledge based systems, one can focus on specific tools, how they can coexist, share representations, etc. While this is important and necessary (and interesting) work, we believe it is necessary also to abstract one level from this activity and consider, not specific sets of tools, but a framework in which tools and knowledge can coevolve *in general.* An initial set of domain independent tools should be provided with such a framework, of course, but new tools arise all through a domain, meta-domain, etc., and such variety must be supported. Abstracting PRL into such a framework is the focus of our current research.

PRL has weaknesses in the particular mechanisms it provides, however we feel the purposes those mechanisms serve, the overall conception of the system, has proved out. We have started designing MetaPrl, the math-free KBM within which systems like PRL might evolve, and will say a few words on this new architecture.

The mathematical base language built into PRL and the code that verifies proofs are being replaced by a constraint language on structures built from definitions. Definitions with no right-side declare primitive forms whose meaning is determined by use (mostly by the constraints). The library will collect up definitions, constraints, and objects (things constructed from definitions) that satisfy the constraints. We think of definitions as frame declarations: those with no right-side are simple frames, those with right-sides are abstracted chunks of larger frame networks that can be examined if the definition is expanded.

The primary reasoning task in MetaPrl is to build objects (frame networks) that satisfy constraints. The framework for chronological backtracking, so natural to use in PRL because of the ML tacticals, is being replaced by a TMS. The TMS is intended to record the (tentative) modifications to objects so that retraction can be performed intelligently. One of the issues we faced in PRL was getting several built-in decision procedures and the many user created tactics to work together effectively. The task was difficult but we had small successes and determined directions for further research. As a result of this experience, additivity and arms-length cooperation have been a major focus of our investigations into a suitable reasoning architecture for MetaPrl.

The idea of the intelligent machine as a medium has started to appear in mainstream AI research. Stefik speaks of a new knowledge medium [5]. However, he does not emphasize the knowledge driven mechanical assistance that we believe is so significant in such a new medium. Lenat et. al. [6] do focus on this, the knowledge *based* aspect of their system, in describing CYC, but they speak only implicitly of CYC as a medium. We believe the explicit focus on both these ideas will lead to fascinating and useful new systems.

References

[1] Bates, J., and Constable, R. *Proofs as Programs*, ACM Transactions on Programming Languages and Systems, Vol 7, No 1 (1985), pp. 113-136.

[2] Constable, R., et al. *Implementing Mathematics with the NuPRL Proof Development Environment*, Prentice-Hall, 1986.

[3] Constable, R., Knoblock, T., and Bates, J. *Writing Programs that Construct Proofs*, Journal of Automated Reasoning, Vol 1, No 3 (1985), pp. 285-326.

[4] Gordon, M., Milner, A., and Wadsworth, C. *Edinburgh LCF: A Mechanized Logic of Computation*, Lecture Notes in Computer Science, Vol 78, Springer-Verlag, New York, 1979.

[5] Stefik, M. *The Next Knowledge Medium*, AI Magazine, Vol 7, No 1 (Spring 1986), pp. 34-46.

[6] Lenat, D., Prakash, M., and Shepherd, M. *CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks*, AI Magazine, Vol 6, No 4 (Winter 1986), pp. 65-85.