Statement of Interest

Bob MacGregor

1989

The current LOOM system includes a term classifier, instance matcher, truth maintainance for both TBox and ABox, default reasoning, full-first-order retrieval language, production rules, pattern-driven methods, a pattern classifier, and automatic detection of inconsistency. Our thesis is that a classification-based architecture will form a good foundation for a next-generation expert system shell. Our primary interests lie in (i) building efficient and comprehensive reasoning systems, and (ii) determining what reasoning facilities, deductive and otherwise, are essential and/or useful for supporting a broad spectrum of applications. Below, we discuss some of our areas of interest.

1 Expressive Power

LOOM philosophy advocates maximizing expressive power, modulo the constraint that the deductive system should do a "competent" (but not necessarily complete) job of reasoning with each language construct. Our informal notion of "competent" might roughly be stated as "If most humans can find a particular kind of logical inference easily, then so should the system."

LOOM implements two distinct languages, a variable-free algebra and a first-order calculus. Concept and relation definitions are phrased in the algebra (primitive operators include :at-least, :all, :range, ...; composition operators include :and, :or, :not, :inverse). Rules can be phrased using either the algebra or the calculus. Currently, default rules must be phrased using a subset of the algebra. The assertion and retrieval languages are based on the calculus, but both contain a means for escaping into the algebra (using the
about operator). We are interested in learning about alternative syntactic
approaches.

NIKL was the first terminological language to include a syntax for defining
binary relations. LOOM has expanded this approach to include escapes
into Lisp and into first-order calculus. We are conjecturing that the LOOM
constructs available for defining relations will meet a significant percentage
of what is needed to define "part-of" and other kinds of domain-independent
relations. We are interested in finding out more about what other people
think is needed in this area.

LOOM implements a feature we call "clipping": When an instance role
marked as single-valued already contains a filler, and then another filler is
asserted for the same role, the first filler is automatically retracted (unless
clipping is disabled). LOOM users have requested a similar facility that
would retract a unary predication when a conflicting unary predication be-
longing to a common disjoint covering is asserted.

LOOM supports two kinds of non-monotonic reasoning: default rules
and closed-world roles (a limited form of circumscription). The default logic
features specificity-based overrides, a "look-ahead" check for constraint vi-
lations, and guarantees a single (possibly non-consistent) extension. The
semantics of defaults represents a trade-off between expressivity and effi-
ciency. Its efficiency is partly dependent on the choice of logic in which it
is embedded (the strict terminological logic), and hence we conjecture that
analyses of default logics that don't consider the interaction between strict
and default rules are likely to miss the mark when it comes time to consider
using them in real applications.

We have supported a partial implementation of closed-world roles long
enough to know that this feature is highly-desirable. It turns out that the
TMS scheme developed to support default rules also works for the closed-
role construct. We are interested in discussing practical approaches to default
and/or circumscription reasoning.

2 Implementation

The LOOM system is heavily optimized (modulo its implementation on top
of CLOS). We consider the performance of the classifier to be of secondary
importance to the performance of the instance matcher and the truth main-
tenance subsystem. We have not done any performance measurements so far, but our guess is that the truth maintenance computations dominate everything else. LOOM implements an object-centered TMS tuned to match the requirements of the classifier. We would be interested in comparing our TMS architecture with other approaches.

The incorporation of defaults required that we convert the strict reasoner from a three-valued to a four-valued semantics, resulting in a seven-valued semantics when defaults are included. We are conjecturing that the default logic we have implemented will admit good performance when the amount of interaction between default rules (in the ABox, not in the TBox) is low. If and when users start building large, dense default rule bases, we may have to "relax" the default semantics (permitting LOOM to perform incomplete truth maintenance on the default rules) if we are to retain acceptable performance. We are interested in discussing other proposals for four-valued semantics, and discussing practical implementations of default reasoning.

We are in the process of adding Horn rules into the system. This change requires that we formally recognize when forward-chaining reasoning occurs, and when backward chaining takes place. Traditionally, users have the option of marking rules as either "forwards" or "backwards." We are taking a different approach — users mark CONCEPTS as either forwards or backwards, and from that vantage point the system will determine in which direction rules are to be processed. We are interested in comparing other approaches to incorporating Horn logic into a terminological architecture.

3 Formal Semantics

When LOOM had a three-valued logic, we had complaints from LOOM users that the introduction of inconsistent facts into the ABox has the nasty habit of spreading incoherence from a bad instance onto neighboring instances. Our solution was to officially introduce a four-valued semantics into the language which acts to prevent inferences whose justifications are based on incoherent instances. We would like to see some attempts to capture formally what the behavior of a system ought to be (i.e., what deductions are explicitly allowed or disallowed) in the presence of an inconsistent knowledge base.
4 Computational Intractability and Undecidability

At ISI we are interested primarily in building systems, rather than developing formal models of how these systems operate. From this perspective, we would like to discuss where we think the important open problems lie, with the hope that some more theoretically-inclined individuals will pursue these avenues for us. We feel that the search for maximally-expressive provably-complete terminological languages ought to be replaced by a search for MEPC's assuming a role-closed database. Bernhard Nebel has achieved what is probably the first result in this direction, proving that the BACK TBox is complete over a role-closed database. Matching against a role-closed database appears to be much easier (from a complexity standpoint) than general subsumption, and we're betting that results analogous to Bernhard's can be pushed much farther.

5 Other Computing Paradigms

The architecture of a term subsumption system (TSS) turns out to be highly compatible with certain other computing paradigms. LOOM provides explicit support for data-driven programming (production rules) and object-oriented programming (pattern-driven methods). By leveraging off of the capabilities of the TSS, we have arrived at versions of these paradigms that are more general (i.e., more powerful) than the standard implementations.

An underlying theme of our paradigms is that of "specificity-based" control. We are interested in exploring the possibilities (and limitations) of using specificity as a control heuristic. For example, should we be reviving specificity as a heuristic for guiding a planner? How successfully can specificity be utilized in place of a more general heuristic, such as the preference scheme used in the SOAR architecture?

With respect to databases, we conjecture that the terminological logics offer a superior alternative to the various types of data models (Entity-Relationship, etc.) commonly employed for specifying database schemata. TSSs also offer superior deductive capabilities, and comprehensive consistency checking. We are interested in learning about what in-roads, if any, TSSs are making into the realm of database technology.