Statement of Interest

Peter F. Patel-Schneider
AT&T Bell Labs
Murray Hill, New Jersey
1989

I have been involved in several theoretical and practical efforts relating to knowledge representation formalisms and systems based on term subsumption languages in the tradition of KL-ONE. I participated in the design and implementation of KANDOR, KRYPTON, and ARGON. I formulated a four-valued semantics for term subsumption languages in an attempt to provide a principled, tractable partial subsumption algorithm. Most recently I have been involved with the design and implementation of CLASSIC.

I am currently interested in:

1. Principled Partiality;
2. Interfaces for Designers;
3. Program Interfaces and Attached Rules and Procedures;
4. The Future of These Systems.

1 Principled Partiality

The intractability results of Brachman and Levesque and others indicated that reasoning in term subsumption formalism was hard, at least in the worst case. However, these results left open the possibility of complete subsumption reasoners that are normal-case tractable, and thus practical. The recent undecidability results by myself and Schmidt-Schauss show that complete
subsumption algorithms for expressively powerful term subsumption formalism do not exist and that we will have to retreat to partial subsumption algorithms.

The problem with partial subsumption is how to characterize the partiality. Because the subsumption algorithm and related algorithms form the central reasoning core of term subsumption formalisms their exact performance is of interest to anyone building general systems using term subsumption formalisms in their KR component. The incompleteness of the term subsumption algorithm must be taken into account when building these systems.

It is not sufficient to just provide a partial subsumption algorithm along with an informal description of how it roughly works; a more complete description is needed, especially if the general system uses attached rules or procedures to perform its processing.

There are several ways of providing descriptions of partial algorithms:

One way is to provide an alternative semantics for the formalism, in which the algorithm are sound and complete. Alternative semantics provides a good characterization of partial reasoning, but only if the alternative semantics can be understood. Devising understandable alternative semantics is a difficult endeavour at best, making this approach less than ideal.

Another way of providing a description of a partial algorithms is via a proof theoretic description of the algorithm. The problem with this sort of description is that the proof theory can be hard to understand. However, for simple subsumption algorithms this approach is probably viable, particularly if it can be extended by adding special-purpose extensions for the inferences missing in particular applications.

Perhaps the best way of providing a description of a partial subsumption or realization algorithm is to present special cases where the algorithm is complete. This is especially useful if it can be shown that the realization algorithm is complete under a large subset of the entire language. (If this is the case then attached rules and procedures will always be applied where appropriate, even if the subsumption hierarchy is not complete.)
2 Interfaces for Designers

Knowledge representation systems based on term subsumption formalisms offer a number of advantages when building large knowledge bases. The subsumption and instance relationships structure the knowledge base in a natural manner. Searching along these relationships allows access to the entire knowledge base without having to know its exact terminology. (Learning the terminology of a large knowledge base is a difficult task.) Term subsumption formalisms, because they contain more than just primitive concepts, are better for this purpose than object-oriented formalisms.

The subsumption hierarchy also provides an excellent way of supporting retrieval (and other tasks such as creation) by reformulation. In this way a knowledge base can be incrementally created without the problem of creating several independent sets of terminology.

Unfortunately, building user interfaces allowing easy browsing and creation of knowledge bases built in term subsumption languages has been hard, largely due to the lack of windowing tools. This situation appears to be easing, and perhaps the time is ripe to explore such interfaces.

3 Program Interfaces and Attached Rules and Procedures

A problem reported by many users of knowledge representation systems built on term subsumption formalisms is the poor interface between the system and other programming languages.

This is not an inherent defect in such systems; in fact, term subsumption formalisms offer a number of advantages for program interfaces. They share with object-oriented formalisms the possibility of hierarchically organizing procedures. They are more expressive than most object-oriented formalisms. This added expressive power allows better description of complicated domains. They perform a type of recognition.

These advantages have been used in the classification-based programming paradigm advocated by John Yen. I would like to see this type of programming extended and integrated into the sort of programming paradigm found in object-oriented systems.
4 The Future

There have been a number of complaints concerning problems encountered when using knowledge representation systems based on term subsumption languages. I feel that there are valid criticisms contained in these complaints. However, I also feel that current knowledge representation systems based on term subsumption languages can play a very important role in so-called knowledge-based systems, i.e., AI systems that need to represent some simple knowledge but mostly need to perform task-specific reasoning.

Extending the reach of knowledge representation systems based on term subsumption languages will require advances in several areas. First, their expressive power will have to be increased. I think that the full power of first-order logic is NOT needed, but that the required level of expressive power will necessitate the use of partial subsumption algorithms. Second, they will have to be extended to incorporate non-term-based reasoning, such as reasoning about parts or causality. I believe that there are a number of domain independent relationships that can be profitably incorporated into these systems. Third, better designer interfaces and better program interfaces will be needed. I think that a combination of rule-based and object-oriented program interfaces will be required.

Bibliography


Peter F. Patel-Schneider. “A Hybrid, Decidable, Logic-Based Knowledge
Representation System". *Computational Intelligence* **3**, 2, May 1987, pages 64-77.

