Statement of Interest

Lori Alperin Resnick
AT&T Bell Labs

September 20, 2001

1989

I have been involved in the design and implementation of the CLASSIC Knowledge Representation system for the past two years. CLASSIC has been developed at AT&T Bell Laboratories. The current members of the project are Ron Brachman, Alex Borgida (a consultant from Rutgers University), Deborah McGuinness, Peter Patel-Schneider, and myself.

The goal of CLASSIC is to improve upon the design of KANDOR, limiting expressive power to maintain tractability, while still being complete in our reasoning. Some of the features of CLASSIC include: partial descriptions for individuals; simple rules; same-as (equality) restrictions on attributes (roles with a min and max of 1); monotonic test restrictions; updates; integration of the host language (i.e., we have defined the host concepts INTEGER, STRING, etc., to be used in restricting fillers for roles); and an open-world assumption.

I have been involved in the design of CLASSIC since its inception, and I have done a large part of the implementation. I have designed and written the error-handling in CLASSIC, so that it returns useful information when an inconsistent individual has been created, to aid the user in correcting the problem. I have implemented the monotonic test restrictions, the modules to handle simple rules, propagation of information, normalization and classification of individuals, subsumption and classification for concepts, the data structures for classic (individuals, concepts, restrictions, symbol tables, graph module, etc.).

During the design and implementation of CLASSIC, we have uncovered a number of interesting issues. Among these are the following:
• We originally had both a FILLS operator (a role on an individual is filled with a particular value) and a CLOSE operator (a role on an individual has no more values) for individuals in our expression language. FILLS and CLOSE are not commutative (you cannot fill a role with a value after that role has been closed). We found that these two operators interacted in interesting ways (i.e., when rules are attached to concepts, a CLOSE can indirectly cause a FILLS), so that depending on what order the internal operations were performed in, either a consistent state would be reached or an error would occur. Thus, we decided to remove the close operator from our language, and only allow the closing of a role as a separate operation.

• We originally had a single type of test function which could be associated with a concept, and which served two purposes. Its first purpose was to determine the integrity of an individual asserted to belong under the concept, and the second purpose was to recognize, during realization, whether an individual belonged under the concept. However, we required that if the individual satisfied the test restriction, and then changed monotonically, it must still satisfy the test restriction. Thus, we found that one type of function did not suffice, since the function which performed integrity checking could not always be restrictive enough to also perform recognition. We now allow two functions: a weaker one for integrity checking, and a stronger one for recognition (although in certain cases, such as for host concepts like INTEGER, we only need a single test, i.e., INTEGERP, since these individuals cannot change).

Because of our desire to remain tractable, and because of the complex interactions between certain features, we currently do not allow the following: same-as (equality) restrictions on multi-valued roles, complete partitioning of concepts, role hierarchies, inverse roles, existentials of the form "at least 1 child who is a doctor".

I’d like to discuss language features and how they interact, and find out what kinds of tradeoffs other people have made in their systems between expressiveness and tractability. We could discuss which features their users have expressed a need for; and how they manage to keep their systems tractable, average-case-tractable, or if they don’t find this to be an important consideration. Our philosophy in CLASSIC is not to try to include every possible
feature we plan to keep it simple, especially since we see CLASSIC as potentially fitting into a larger system (i.e., it might serve as the working memory for an OPS-like rule system, with a database as the back end).

I would be interested in discussing some implementation issues, including how people handle errors and return useful information to their users; classification and subsumption algorithms; storage space required vs. speed of retrieval when implementing the relationships in the knowledge base and inherited information, etc.

There are some issues on updates/retractions I’d be interested in discussing:

- which information can the user retract (i.e., only information told by the user);

- what is the result of removing an individual from a class (i.e., if an individual I is removed from PERSON, does it revert to MAMMAL, even if the user didn’t specifically assert MAMMAL(I), or does it in this case revert to THING?);

- How do people implement updates (i.e., with dependency lists, or a TMS).