PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1992,
edited by ICOT. © ICOT, 1992

Knowledge Representation Theory Meets Reality: Some Brief Lessons from the CLASSIC Experience

Ronald J. Brachman

AT&T Bell Laboratories, 600 Mountain Ave.,
Murray Hill, NJ 07974-0636, U.S.A.
rjb@research.att.com

Abstract

Knowledge representation is one of the keys to Artificial Intelligence, and as a result will play a critical role in many next generation computer applications. Recent results in the field look promising, but success on paper may be misleading: there is a significant gap between a theoretical result or proposal and its ultimate impact in practice. Our recent experience in converting a fairly typical knowledge representation design into a usable system illustrates how many aspects of "reduction" to practice can significantly influence and force important changes to the original theoretical foundation. I briefly motivate our work on the CLASSIC representation system and outline a handful of ways in which practice had significant feedback to theory. The general lesson for next generation applications is the need for us in our research on core technology to take more seriously the influence of implementation, applications, and users.

1 Knowledge Representation

Representation of knowledge has always been the foundation on which research and development in Artificial Intelligence has rested. While no single representation framework has come to dominate the field, and while there are important challenges to the utility of conventional representation techniques from "connectionists" and others, it is very likely that the next generation of AI and AI-related applications will still subscribe to the hypothesis that intelligent behavior can arise from formal reasoning over explicit symbolic representations of world knowledge.

The centrality of the need to represent world knowledge in AI systems, expert systems, robots, and Fifth Generation applications has helped increase interest in formal systems for representation and reasoning—so much so that over the last decade, the explicit subfield of "Knowledge Representation" (KR) has taken on its own identity, with its own international conferences, IFIP working group, etc. This subfield has been prolific. It has attracted the attention of the greater AI community with highly visible problems like the "Yale Shooting Problem" and systems like CYC. It has collected its own set of dedicated researchers, and has increasing numbers of graduate students working on formal logics, nonmonotonic reasoning, temporal reasoning, model-based diagnosis, and other important issues of representation and reasoning.

It is probably fair to say that in recent years, formal and theoretical work has become preeminent in the KR community.[1] Concomitantly, it appears to be generally believed that when the theory is satisfactory, its reduction to practice will be relatively straightforward. This transition from theory to practice is usually considered uninteresting enough that it is virtually impossible to have a technical paper accepted at a conference that addresses it; it seems to be assumed that all of the "hard" work has been done in developing the theory.

This attitude is somewhat defensible: it is common in virtually all other areas of AI; and there often really isn't anything interesting to say further about a KR formalism as it is implemented in a system. However, my own group has had substantial recent experience with the transition of a knowledge representation system from theory to practice that contradicts the common wisdom, and yields an important message for KR research and its role in next generation applications. In particular, our view of what we thought was a clean and clear—and "finished"—formal representation system was substantially influenced by the complexity and constraint of the process of turning the logic into a usable tool.

2 The CLASSIC Effort

As of several years ago, we had developed a relatively small, elegant representation logic that was based on many years of experience with description hierarchies and a key inference called *classification*. As described in a companion paper at this conference [Brachman *et al.*, 1992], the CLASSIC system was a product of many years of effort on numerous systems, all descended from the KL-ONE knowledge representation system. Work on KL-ONE and its successors grew to be quite popular in the US and Europe in the 1980's, largely because of the semantic cleanliness of these languages, the appeal of object-centered (frame) representations, and their provision for some key forms of inference not available in other formalisms (e.g., description classification). The reader familiar with KR research will note that numerous publications in recent years have addressed formal and theoretical issues in "KL-ONE-like" languages, including formal semantics and computational complexity of variant languages. However, the key prior efforts all had some fundamental flaws, and work on CLASSIC was in large part launched to design a formalism that was free of these defects.

Another central goal of CLASSIC was to produce a compact logic and ultimately, a small, manageable imple-

[1]This has happened for numerous reasons, and while it may have some negative consequences (as addressed here), it is positive in many respects. The early history of the field was plagued by vague and inadequate descriptions of ad hoc solutions and computer programs; recent emphasis on formality has encouraged more thorough and rigorous work.

mented representation and reasoning system. A small system has important advantages in a practical setting, such as portability, maintainability, and comprehensibility. Our intention was to eventually put KR technology in the hands of non-expert technical employees, to allow them to build their own domain models and maintain them. CLASSIC was also designed to fill a small number of application needs. We had had experience with a form of deductive information retrieval (most recently in the context of information about a large software system [Devanbu *et al.*, 1991]), and needed a better tool to support this work. We also had envisioned CLASSIC as a deductive, object-oriented database system (see [Borgida *et al.*, 1989]; success on this front was eventually reported in [Selfridge, 1991]).

After analyzing the applications, assessing recent progress in KL-ONE-like languages, and solving a number of the technical problems facing earlier systems, we produced a design for CLASSIC that felt complete; the logic was presented in a typical academic-style conference paper in 1989 [Borgida *et al.*, 1989]. In this design, some small concessions were made to potential users, including a procedural test facility that would allow some escape to the host implementation language for cases that CLASSIC could not handle. Given the clarity and simplicity of this original design of CLASSIC, we ourselves held the traditional opinion that there was essentially no research left in implementing the system and having users use it in applications. At that point, we began a typical AI programming effort, to build a version of CLASSIC in COMMON LISP.

3 Influences in the "Reduction" to Practice

As the research LISP version neared completion, we began to confer with colleagues in a development organization about the potential distribution of CLASSIC within the company. Despite the availability of a number of AI tools in the marketplace, an internal implementation of CLASSIC held many advantages: we could maintain it and extend it ourselves, in particular, tuning it to real users; we could assure that it integrated with existing, non-AI environments; and we could guarantee that the system had a well-understood, formal foundation (in contrast to virtually all commercially available AI tools). Thus we undertook a collaborative effort to create a truly practical version of CLASSIC, written in C. Our intention was to develop the system, maintain it, create a training course, and eventually find ways to make it useful in the hands of AI novices.

To make a long story short, it took at least as much work to get CLASSIC to the point of usability as it did to create the original logic that we originally thought was the culmination of our research. Our view of the language and knowledge base operations supporting it changed substantially as a result of this undertaking, in ways that simply could not be anticipated when consider a paper design of the logic.

The factors that influenced the ultimate shape of CLASSIC were quite varied, and in most cases, were not influences that we—or most other typical researchers, I suspect—would have expected to have forced more research before the logic was truly finished. These ranged from the need to be reasonable in the release and maintenance of the software itself to some specific needs for key applications that could not really have been anticipated until the system was actually put into practical use. Here is a brief synopsis of the five main types of issues that influenced the ultimate shape of the CLASSIC system:

- the constraints of *creating and supporting a system for real users* caused numerous compromises. For one thing, upward compatibility of future releases is a critical issue with real software, and it meant that any construct in the language in which we were not completely confident might better be left out of the released system. Issues of run-time performance (which also dictated the exclusion of some features) also had surprising effects on what we could realistically include in the released version.

- certain detailed *implementation considerations* played a role in determining what was included in the system. These included certain tradeoffs that affected the design, such as the tremendous space consequences an inverse relationship ("inverse roles") feature would have had, or the consequences of certain fine-grained forms of truth maintenance (to allow for later retraction of asserted facts). Some features (our **SAME-AS** construct, for example) were just so complex to implement that they were better left out of the initial release.

- *concern for real users* alerted us to issues easily ignored with a pure logic. These involved the sheer learnability and usability of the language and the system. Error-handling, for example, was of paramount concern to our real consumers, and yet the very idea never arose when considering the initial CLASSIC language. Similarly, the uniformity of abstractions and the simplicity of the interface were critical to acceptability of our system. The potential consequences of user "escapes" with side-effects was another related concern. Finally, explanation of the system's behavior—again, not an issue when we designed the logic—might make the difference between success and failure in using the system.

- as soon as a system is put to any real use, mismatches in its capabilities and *specific application needs* become very evident. In this respect, there seems to be all the difference in the world between the few small examples given in typical research papers and the details of real, sizable knowledge bases. In the case of CLASSIC, our lack of attention to the details of numbers and strings in the logic meant substantial more work before implementation. Another issue that plagued us was the lack of attention to a query language for our KR system (a common lack in most AI KR proposals).

- finally, *what looked good (and complete) on paper* did not necessarily hold up under the fire of real use. Even with a formal semantics, certain operators prove tricky to understand in practice, and subtle interactions between operators that arise in practice are rarely evident from the formal work. Simply being forced by an implementation effort to get every last detail right certainly caused us to re-examine several things we thought we had gotten correct in the original logic, and I suspect this would be the case with virtually every sufficiently complex KR logic that ends up being implemented.

4 Some Lessons

The main lesson to be learned here is that despite the ability to publish pure accounts of logics and their theoretical properties, the true theoretical work on knowledge representation systems is not really done until issues of implementation and especially of use are addressed head-on. The "theory" can hold up reasonably well in the transition from paper to system, but the typical KR research paper misses many make-or-break issues that determine a proposal's true value in the end. Arguments about needed expressive power, the impact of complexity results, the naturalness and utility of language constructs, etc., are all relatively hollow until made concrete with specific applications and implementation considerations.

For example, in our context, the right decision was clearly to start with a small version of the system for release, and extend it only as needed. Given the complexity of software maintenance, it may never make sense to try to anticipate in advance all possible ways that all possible users might want to express concepts.[2] A small core with an extension mechanism might in reality be better than a large, extraordinarily expressive—and complex—system. In the case of CLASSIC, we have been able to place in the hands of relatively naive users a fairly sophisticated, state-of-the-art inference system with a formal semantics and well-founded inference mechanism, and have them use it successfully, needing only to make a small number of key extensions to meet their real needs.

There are several consequences here for next generation applications of knowledge representation research. First, it is important that the research community recognize as legitimate and important the class of issues that arise from implementation efforts—issues relating to size, for example, that have always been the legitimate concern of the database community; issues relating to implementation tradeoffs and complexities; and issues relating to software release and maintenance. Second, unless our KR proposals are put to the test in real use on real problems, it is almost impossible to assess their real value. So much seems to be different when a proposal is reduced to practice that it is unclear what the original contribution really is. Third, it is quite critical that at least some fraction of the community address directly the needs of users and the constraints and issues in their applications. Too much research with only mathematics as its driving force will continue to lead KR (and other areas of AI research) farther afield. Not only that, it is clear that truly interesting research questions arise when driven from real rather than toy or imagined needs.

References

[Borgida *et al.*, 1989] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: A Structural Data Model for Objects. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 59–67, June 1989.

[Brachman *et al.*, 1992] R. J. Brachman, A. Borgida, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Resnick. The CLASSIC Knowledge Representation System, or, KL-ONE: The Next Generation. In *Proceedings of the International Conference on Fifth Generation Systems*, Tokyo, June 1992.

[Devanbu *et al.*, 1991] P. Devanbu, R. J. Brachman, P. G. Selfridge, and B. W. Ballard. LaSSIE: A Knowledge-Based Software Information System. *CACM*, 34(5):34–49, May 1991.

[Selfridge, 1991] P. G. Selfridge. Knowledge Representation Support for a Software Information System. In *Proceedings of the Seventh IEEE Conference on AI Applications*, pages 134–140, Miami Beach, Florida, February 1991.

[2]Ironically, the ongoing and sometimes virulently argued debate over how much expressive power to allow in KR systems may in the end be settled by simple software engineering considerations.