Artificial Intelligence 61 (1993) 95–104
Elsevier

ARTINT 941

Book Review

D.B. Lenat and R.V. Guha, *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project* *

John F. Sowa

IBM Systems Research, 500 Columbus Avenue, Thornwood, NY 10594, USA

Revised February 1992

The Cyc project, started by Doug Lenat at MCC in 1984, is the most ambitious knowledge representation project ever undertaken. It embodies Lenat's current ideas for a system intended to encode all of commonsense knowledge. By the year 1999, he hopes that "no one would even *think* of buying a computer that doesn't have Cyc running on it". The book by Lenat and Guha is a report on the project as it was in 1989. A review of that book must distinguish four different things: the book itself, the Cyc project as it was when the book was written, the Cyc project today, and the developments that the designers are planning for the future. Of these four, the last two are probably the most interesting.

This review has been difficult for me to write, because my thoughts about Cyc have changed a great deal since I first read the book in the spring of 1990. Doug Skuce showed me a copy of his review (which also appears in this issue), and it is similar to what I had originally intended to write. I agree with his complaints about the confusing organization of the book and the lack of precise definitions. Despite our reservations, we both used

Correspondence to: J.F. Sowa, IBM Systems Research, 500 Columbus Avenue, Thornwood, NY 10594, USA.

* (Addison Wesley, Reading, MA, 1990); 372 pages.

the Cyc book as a text for courses in knowledge representation. We chose it for two reasons: first, in pushing knowledge systems technology beyond any previously established levels, the Cyc project is uncovering many new problems and phenomena; second, the Cyc book discusses serious issues in knowledge representation, unlike many books that are little more than a manual on how to write rules for a particular expert system shell.

1. Cyc as presented in the book

I used the book as a supplementary text in a one-week course on topics in knowledge acquisition and representation. In the course, three different instructors each lectured for three half-day sessions. All of the students had taken previous courses on AI, and some had a considerable amount of practical experience in implementing expert systems in Prolog and various shells. I spent half a day lecturing about the Cyc project and went into enough detail on the Cyc language to enable the students to read the book on their own. The students had mixed reactions. Most of them were intrigued by the grand goals of the Cyc project, but they would have preferred a less ambitious system they could use immediately to a grandiose system that might solve all the problems of AI at some indefinite time in the future. In my lectures, I tried to emphasize issues in ontology and some of the problems of representing knowledge about time, change, processes, events, and the distinction between objects and stuff. Although the book addresses these topics, the students found it hard to distinguish the general principles from the details of the Cyc language. As a language, the Cyc frame notation is readable, but the constraint language is a rather ugly version of predicate calculus with Lisp-style parentheses and symbols like #%ForAll and #%LogImplication. I agree with Skuce that the Cyc book is less than ideal as a text for a course on "building large knowledge-based systems", but it is useful as supplementary reading.

As documentation on Cyc, the book is a good report on the state of the project in 1989. But the changes in my attitude have resulted from recent lectures and demos and from discussions with Lenat, Guha, and other project members. Before getting into the new developments, I should start with a list of my original reservations:

(1) When I first read the Cyc book, I had just finished writing a paper on "knowledge soup" [8]. The theme of that paper is that the total knowledge in a person's head is too large and too disorganized to be called a knowledge base. Instead, it consists of many chunks of knowledge, each of which is internally consistent, but possibly inconsistent with one another. The metaphor of knowledge soup more

accurately conveys the highly fluid, loosely organized, and rapidly changing state of human knowledge. I presented that paper at a workshop where it found a sympathetic response, especially from the participants who had extensive experience in building large knowledge bases [7]. With that background, I was skeptical of the emphasis in Cyc on maintaining global consistency and the absence of any mechanism for partitioning it in smaller chunks. The goal of global consistency not only seemed unattainable, it also seemed too inflexible to represent commonsense knowledge.

(2) The idea of using commonsense knowledge as a starting point for building more advanced knowledge bases also seemed dubious at best. On any particular topic, an expert and a layman would probably differ on almost every point, and the differences are most striking in the foundations. The background knowledge of a typical secretary or bus driver, for example, would not make a useful starting point for an expert system to diagnose bacterial infections, configure computer systems, or explore for oil deposits. For practical applications, the common knowledge shared by every high school graduate is so superficial compared to an expert's that it would have to be completely rewritten. Lenat claims that common sense might also be useful as a supplement to an expert's deeper knowledge. That may be true, but the Cyc project as presented in the book makes no provision for different, possibly inconsistent viewpoints on the same subject matter.

(3) I was also concerned that the Cyc ontology might be so deeply buried in their knowledge base that it couldn't accommodate new ideas that might come out of research in AI, linguistics, and philosophy. As the knowledge soup paper concludes, no ontology can ever be complete and perfect for all time. And like Skuce, I had reservations about the lack of perfection in the presentation of the Cyc ontology and the paucity of references in their bibliography.

(4) When I started to write this review, I had a long discussion about Cyc with Bob MacGregor, who spent many years working on classification systems in the KL-ONE tradition [5]. One of the most characteristic features of those systems is the deliberate restriction of expressive power in order to improve performance. Yet the Cyc language provides the full expressive power of first-order predicate calculus with sets. MacGregor doubted that a system with that much expressive power could perform well with current theorem-proving technology. Lenat claims that Cyc's special-purpose inference mechanisms somehow enable it to achieve good performance. But the trade-offs between expressive power and performance are glossed over with little or no discussion.

(5) These objections would be moot if Lenat could show some practical applications built on top of Cyc. But none are even mentioned in the book. The MYCIN project at Stanford and the R1 project at Carnegie Mellon both developed impressive applications in less time with a smaller team and less money. Lenat said that he wouldn't be able to demonstrate such results until he built up a critical mass of background knowledge. Yet he still could have tested his approach by picking a narrow topic and encoding enough knowledge about the domain to show that Cyc would really work.

(6) Finally, many people have raised serious doubts about the whole enterprise of knowledge representation. The connectionists claim to have a better way. Phenomenologists like Hubert Dreyfus [1,2] have never believed that true AI was possible, and they have won converts among AI researchers like Terry Winograd [9]. Even advocates of logic like Drew McDermott [6] have expressed doubts. Although I firmly believe in the importance of logic and symbolic knowledge representations, I agree with many of the critics that image-like or analog simulations may be better suited to certain kinds of reasoning. Much if not all of that reasoning might be simulated on digital computers, but it would probably require more numeric computation than logical inference. A true commonsense reasoning system would have to combine symbolic and analog computation in ways that are just beginning to be explored in research projects.

These criticisms reflect my opinion of Cyc in early 1990. In September of that year, I visited the Cyc project and began a series of discussions with Lenat and other project members. Although the Cyc project has not solved all these problems, they have made major changes in the project goals and directions.

2. New developments in Cyc

My opinion of Cyc began to improve when I ignored the word "commonsense". Although Lenat still uses it, I believe that more mundane applications of Cyc are more promising. One of the current demos, for example, is a car sales adviser that shows how Cyc can access independent knowledge sources and integrate them with the inferencing. It asks questions about a customer's needs and interests and suggests car models to consider. During the consultation, Cyc accesses several different databases to get information: one is a relational database using SQL that has the Kelly *Blue Book* data on prices; another is an object-oriented database using ORION that has information about options and packages; and a third is the *Consumer Reports* database

on reliability and customer satisfaction. Instead of doing deep inferencing, Cyc serves as an intelligent front-end to independently developed databases, each of which may have a totally different format. Lenat is surprised that people are more excited by Cyc's ability to access external databases than by the many wonders of AI that he has built into it. But large database systems are the mainstay of the computer industry, and communication between them is limited. The sales division and the manufacturing division of a company, for example, both deal with the same products; but their databases are usually incompatible with one another. If Lenat could demonstrate how Cyc could help to integrate and access independently developed databases, that feature alone could make it a commercial success.

The innovation that allows Cyc to access independent knowledge sources is the result of Guha's Ph.D. dissertation at Stanford under the direction of John McCarthy [3]. He introduced the notion of "microtheories" as a basis for reorganizing and partitioning Cyc's knowledge base into more manageable chunks. That kind of partitioning is consistent with my knowledge soup paper, which proposed "a two-level structure: a large reservoir of loosely organized encyclopedic knowledge, called *knowledge soup*; and floating in the soup, much smaller, tightly organized theories that resemble the usual microworlds of AI". Each microtheory contains a few axioms or rules that could be carefully polished and tested. Then new applications could be built by assembling multiple microtheories and integrating them with "articulation rules" that allow one theory to use results from another. The microtheories could be implemented in Cyc's native language, or they could be knowledge sources from external programs, databases, and expert systems.

In May 1991, Lenat discussed some of the recent extensions to Cyc in a talk at IBM Research in Yorktown. Following is an excerpt from his abstract:

> How does the translation between the EL (epistemological level) and HL (heuristic level) work? What specifically forced us to augment, and eventually all but abandon, frames as a representation? How exactly does argumentation work in Cyc? What is the nature and internal structure of Cyc's microtheories, and how is lifting (articulation, importing) done? What sort of metalevel representation and reasoning is available, and how is it kept efficient? What knowledge has been entered in the past year? How are knowledge enterers trained; what is the "script" for entering knowledge into Cyc? How are they and the knowledge base kept from semantically clashing or diverging over time?

Most of the material in that lecture was either omitted from the Cyc book or mentioned only briefly. Microtheories, for example, are probably the most

important innovation, but they weren't part of the system when the book was written. The distinction between the purely declarative EL and the more procedural HL is mentioned on pages 52 and 53, but it is never emphasized or discussed in detail. Most of the examples in the book are represented in the frame language, which has been "all but abandoned" in favor of the constraint language.

Besides making the knowledge base more modular, the microtheories allow different knowledge engineers to work independently without interfering with one another. Even more importantly, they allow a new expert system to be created by picking and choosing from the predefined microtheories. The car sales adviser, for example, could be developed by starting with a microtheory about cars, a microtheory about buying and selling, and a microtheory about consumers and their preferences. Small, reusable microtheories could serve as building blocks in the same way as objects in an object-oriented language. External knowledge bases and databases running on remote systems could also be treated as microtheories within the overall Cyc framework. In his lecture, Lenat said that there were several ways in which external systems could be integrated with Cyc:

(1) The simplest way is to access an external application as a passive knowledge source. Cyc would call it to get data when needed.
(2) A higher level of integration is to treat an external application as a Cyc microtheory. Cyc might have knowledge about a broad area, such as computer disk drives, and an external expert system might be used to diagnose failures of a particular device type. In this way, Cyc could cooperate with external programs in much the same way that it would cooperate with a microtheory implemented in the native Cyc Language.
(3) The highest level of integration is to treat Cyc as an upward-compatible follow-on to an existing expert system shell. The Cyc framework supports multiple inference engines, each specialized for different reasoning strategies and each with its own kinds of heuristic rules. New inference engines could be added to Cyc to support rules that were written for older systems. Then an application could be migrated to Cyc with a minimal amount of change, and further extensions could take advantage of the knowledge and inference facilities in the full Cyc system.
(4) Another way of using Cyc in conjunction with other tools is to treat it as a resource for knowledge acquisition. For example, if one wanted to build a car sales adviser in another expert system shell, the microtheories for cars, buying and selling, and consumer preferences could be merged, tested, and debugged in Cyc. Then the purely declarative aspects could be translated to the other knowledge

representation language. Finally, a knowledge engineer could add the heuristic rules in the target language to make it a usable expert system.

The demo of the car sales adviser illustrates the first level of integration; Guha's implementation of microtheories supports the second level. The third and fourth levels lead to important research and development issues, and Cyc could be a valuable tool for exploring new ways of handling them.

Lenat also said that he has experienced a major change in his own way of thinking and working over the years. When the Cyc project began in 1984, he had a very "scruffy" outlook with an emphasis on frames and procedural code. But as their knowledge base grew, they kept running into problems of consistency and maintainability. To solve those problems, they gradually cleaned up the interfaces and made a sharp distinction between the logic-based EL and the more procedural HL. Many people have criticized frames as too limited to support all of logic: anything that can be said in frames can also be said in first-order logic, and the more interesting kinds of knowledge cannot be expressed in frames at all. As a result of their experience in knowledge encoding, Lenat and his colleagues have reduced their dependence on frames. They still use them as an optional notation for simple information, but the frames are now treated as just a special case of first-order logic. Today, Lenat says that he has almost become a "neat" with a very logic-oriented approach. However, he emphasized that every one of the changes towards logic was adopted for practical reasons of making the system more powerful, reliable, or extendible.

The earlier version of Cyc supported certainty factors in the style of fuzzy logic and many expert systems. But they found that the certainty factors were largely arbitrary, no one had a good theory about how to determine the "best" values, it became very hard to keep them consistent when multiple knowledge engineers were adding information, and spurious inferences were creeping in that resulted from the arbitrary values. Now, they have dropped the numeric-valued certainty factors in favor of five distinct levels:

- definitely true,
- true by default,
- unknown,
- false by default,
- definitely false.

To allow an ordering of defaults, they use metalevel statements that one default is more likely than another, but they do not assign numeric certainty factors to them. By dropping certainty factors, they have taken another step towards a pure logic-based system.

3. Directions for the future

The new developments in Cyc have not completely answered all my earlier objections, but they have satisfied me that Cyc has already made important contributions to AI and should continue to enrich both theory and practice. Following is my current reassessment of the six objections I discussed in Section 1:

(1) The partitioning of Cyc into microtheories is probably the most important development for organizing the knowledge soup. But it also raises new research issues: How are the microtheories related to one another, how are they combined or modified to form new theories, and how does the system choose which theory to use to answer any particular question? For natural language understanding, the partitioning into microtheories has both strengths and weaknesses. For highly restricted natural language, as in database query and specialized sublanguages [4], the partitioning could be helpful, especially if there is a one-to-one mapping between microtheories and sublanguages. For less restricted language, as in machine translation and information retrieval, a single sentence might use terms from several different microtheories. The partitioning could then make semantic analysis more difficult, especially if the system has to search through many microtheories to find the correct word sense.

(2) The Cyc project has not convinced me that commonsense knowledge is a useful foundation for building highly specialized expert systems. But its broad, shallow knowledge base might enable Cyc to classify the deeper knowledge embedded in specialized microtheories. With such a structure, it could behave like a librarian: it wouldn't be able to answer a detailed question about anything, but it would know enough about everything to be able to direct the question to a microtheory that might be able to answer it. With a large library of microtheories, Cyc could also serve as a resource for building new systems by reusing and combining microtheories.

(3) The partitioning in microtheories should enable Cyc to accommodate different ontologies. It could also permit finer distinctions in a microtheory than in the broader knowledge base. Lenat gave an example of a physician who would carefully distinguish symptoms from diseases, while a layman might confuse them by saying that someone suffered from a cough or suffered from a cold. A microtheory for the physician might be much richer than a microtheory for the patient; but to enable them to communicate, they would need a shared ontology of common terms like *sniffle*, *cough*, and *fever*. The framework of microtheories is a promising approach to knowledge sharing

between systems with different ontologies, but a lot of research and development is needed to make it work.

(4) The partitioning could also answer some of MacGregor's objections, since different microtheories could use different reasoning strategies tailored for different kinds of problems. At the broadest level, most of the reasoning would be simple retrieval, which could be done in sublinear time with good indexing techniques. Deeper reasoning would only be done within smaller microtheories. But a great deal of experimentation is needed to determine what kinds of applications are best suited to different reasoning strategies and how they can all cooperate in an integrated system.

(5) The new demos make Cyc look like a robust system for serious applications. The car sales adviser suggests that it has great potential as an intelligent front-end for external databases and applications. But to make a convincing case, the developers must show that these demos can grow into real applications in production use.

(6) Finally, the question of whether Cyc could ever achieve true common sense at a human level is an interesting theoretical issue, but it's irrelevant for many applications. Database systems, for example, are not at all human-like in their ability to store and dispense large volumes of knowledge. In fact, their very nonhuman characteristics are what make them so valuable. But their lack of human friendliness makes them hard to learn and use. Some amount of background knowledge might help to make them friendlier and more flexible, even if it was far from a truly human level of common sense. If Cyc could become a general help facility that could organize knowledge about any application running on a computer, that would be sufficient to realize Lenat's hopes that no user would want to be without it.

Most of this review has been directed at topics that are missing from the Cyc book. In the preface, the authors promise more to come in a 1994 edition. That should be much more interesting, but the 1990 book is still an important source for anyone who is doing research and development on advanced AI systems. For those who merely want hints on how to use an existing expert system shell, however, this book may be a disappointment.

References

[1] H.L. Dreyfus, *What Computers Can't Do* (Harper & Row, New York, 2nd ed., 1979).

[2] H.L. Dreyfus, Introduction, in: H.L. Dreyfus, ed., *Husserl, Intentionality, and Cognitive Science* (MIT Press, Cambridge, MA, 1982) 1–27.

[3] R.V. Guha, Contexts: a formalization and some applications, Tech. Rept. ACT-CYC-423-91, MCC, Austin, TX (1991).

[4] R. Kittredge and J. Lehrberger, eds., *Sublanguage: Studies of Language in Restricted Semantic Domains* (de Gruyter, New York, 1982).
[5] R. MacGregor, The evolving technology of classification-based knowledge representation systems, in: J.F. Sowa, ed., *Principles of Semantic Networks* (Morgan Kaufmann, San Mateo, CA, 1991) 385–400.
[6] D.V. McDermott, A critique of pure reason, *Comput. Intell.* **3** (1987) 151–160.
[7] B.G. Silverman and A.J. Murray, Full-Sized Knowledge-Based Systems Research Workshop, *AI Mag.* **11** (5) (1991) 788–794.
[8] J.F. Sowa, Crystallizing theories out of knowledge soup, in: Z.W. Ras and M. Zemankova, eds., *Intelligent Systems: State of the Art and Future Directions* (Ellis Horwood, New York, 1990) 456–487.
[9] T. Winograd and F. Flores *Understanding Computers and Cognition* (Ablex, Norwood, NJ, 1986).