Teaching Knowledge Representation: Challenges and Proposals

Leora Morgenstern
IBM T.J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532
leora@watson.ibm.com

Richmond H. Thomason
AI Laboratory
University of Michigan
Ann Arbor, MI 48109–2210
rich@thomason.org

Abstract

This position paper was prepared for a panel to be held at KR2000. The authors hope that it will help to stimulate discussion, planning, and cooperation concerning teaching issues in the KR community.

1 INTRODUCTION AND MOTIVATION

Knowledge Representation has played a crucial role in the development of Artificial Intelligence, and remains one of the strongest subfields of AI. From the earliest days of AI, leading researchers such as (McCarthy 1959), (McCarthy & Hayes 1969), and (Minsky 1969) have argued that in order for a program to act intelligently, it must have sophisticated methods of representing and reasoning with knowledge. The contributions of KR research—e.g., the use of formal logic for representing knowledge, automated theorem proving techniques, logic programming, semantic networks, and inheritance techniques—have been at the forefront of the AI intellectual scene. Whatever one may say about the value of human-level AI as a research goal,[1] KR continues to be a crucial source of ideas for AI technology. It has influenced the activity at AI research and development laboratories, and many of its ideas have spread to the general computer science community. Among the best known examples of this phenomenon are object-oriented programming languages (such as C++ and Java), to which inheritance techniques are central, and program specification languages which are modelled after Prolog.

Despite the centrality of KR to AI and Computer Science, it remains somewhat marginalized in the CS and AI curriculum. KR is rarely required in the CS curriculum; indeed, both graduate and undergraduate AI students can and often do complete their studies without taking a course in KR. Many universities don't even offer a course in KR, although departments which offer a two-semester course in AI often teach a good deal of KR in the second semester. There are few textbooks in KR, although there are KR-related texts like (Davis 1990) and (Sowa 1999) that concentrate on specific approaches and subtopics.

KR's lack of visibility in the curriculum is bad for CS and AI students, bad for CS departments, and bad for the future of KR itself.

A good KR course can be an intellectual eye-opener. It offers students an excellent—often, a unique—opportunity to see how ideas from computer science interact with system development. Typical CS students come to their first AI course with good programming skills and a fair amount of experience in small-scale projects. But without KR, they are unlikely to understand the importance of analyzing the reasoning task at hand, of separating out declarative elements, or of modularizing system design. KR not only serves as an important subject in its own right, introducing the research topics covered in KR, but often provides the transition from a naive to a more sophisticated understanding of system design.

KR applications abound with illustrations of this point. In developing an expert system, it is important to separate out the KR issues from software engineering issues. When designing educational software, it is crucial to correctly model the underlying domain knowledge so that one can construct coherent and useful explanations for the users. In developing a large-scale knowledge base, one must tackle head-on issues of ontology development to ensure proper organization for efficient updating, retrieval, and inference.

KR provides a way of integrating formal representa-

[1] See (Nilsson 1995).

tion languages and ideas from theoretical CS with challenging applications. This point has been stressed by (Sandewall 2000), who argues that the role of formal representation in AI and CS is akin to

> the role of calculus in engineering, that is, as a conceptual and notational tool that allows one to model phenomena in the world with precision, and to design software systems based on those models.

Applications and programming systems change at a relatively rapid rate, and are often best learned in the workplace. Basic scientific principles, on the other hand, have a longer horizon, and if they are not learned in an academic setting are unlikely to be acquired in the workplace. Computer *science* as a field presupposes that there are such basic principles, and that they are crucially important for computing technology. As one of the best ways of illustrating the importance of scientific ideas in software development, KR is important for a balanced and successful CS curriculum.

Anyone attending a KR meeting or reading a KR Proceedings doesn't need to be convinced of these things. But it may be important for the KR community to hone these arguments and practice them on colleagues, in an effort to secure a more favorable environment for teaching KR in universities. We need better publicity for the importance of KR in the curriculum, as well as better solutions to the challenges of teaching KR.

In the next sections, we first examine some of the major challenges for KR teachers, and propose some solutions; then we discuss issues of content and resources.

2 CHALLENGES OF TEACHING KR

Teaching KR can be an incredibly uplifting experience. There is great satisfaction in introducing students to a fundamentally different way of thinking about what they do—and KR, with its emphasis on formal methods of representation, does just that. At the same time, anyone who has taught KR on a regular basis probably has experienced some problems in the process. There are many potential difficuties facing KR teachers. Below, we single out four major challenges. We then discuss the relation between teaching KR and training users in the commercial world to use KR-based applications. The connection between these two activities suggests that developing improved methods for teaching KR can also facilitate the dissemination of KR within the commercial arena.

2.1 LACK OF STUDENT PREPARATION

Because the central idea of KR is the representation of knowledge in formal languages, students need a good grasp of logic to succeed in the study of KR. Specifically, students need to know first-order logic, the most important general-purpose representation and reasoning system. They also need to know basic metatheoretic techniques, including model theory and proof theory, preferably through a proof of the soundness and completeness of first-order logic. Moreover, they should be competent in translating at least simple natural language sentences to first-order logic and should have some understanding of the difficulties that translation from natural language to logic can present. Ideally, they should have some familiarity with other logics, such as modal, temporal, and dynamic logics; certainly first-order logic is the barest requirement.

Unfortunately, in our experience, many students come in with a smattering of logic, picked up, sometimes reluctantly, in a required introductory logic course somewhere along the way. Logic is seldom taught as a separate unit in the Computer Science curriculum, and content and standards of logic instruction differ widely outside of computer science. Many introductory courses concentrate mostly on propositional logic, and give students only a taste of first-order logic. Many courses are weak on metatheory. Students rarely learn how to translate between natural language and a formal language like first-order logic.

Meeting the Challenge:

There is no simple cure. It is clear that in the long term we ought to agitate for more training in logic, and for this training to occur early on in a student's career. High school, or even junior high school, is not too early to introduce students to both propositional and predicate logic. Logic should be emphasized in college as an important part of the core curriculum, and certainly should be required early on for any student majoring in computer science.

We don't imagine that waging this sort of campaign will be easy. But we should use the persuasive arguments that have been made by, among others, Robert Kowalski (Kowalski 1990, 1993), who has spoken and written at length about the close connection between writing clearly and being able to translate natural language into first-order logic. Kowalski argues that writing and rewriting one's thoughts in successively clearer drafts of English (or one's native natural language) brings one closer to the process of formalizing these ideas in logic. Making explicit and publicizing the close connection between training in logic and the

development of writing skills might help elevate logic's importance in the curriculum—not only in the CS curriculum but in the liberal arts and sciences curriculum. Our motto ought to be that training in logic develops strong and flexible minds; it should become a core part of the twenty-first century curriculum.

This is an ambitious goal, which will take time to implement. In the short term, we need quick solutions to the problems that arise when students are unprepared. Most KR teachers have probably experienced the fallout (both in terms of student attrition and students' complaints) from assigning the first homework requiring substantial formalization in logic. It is usually at this point that students realize how poorly prepared they are. How can we help them? We have four suggestions.

- *Mini-courses.* It is best if students realize quickly that they have to make up deficiencies; possibly this can be accomplished by making assignments available before class begins or on the first day of class. To bring students up to speed, intensive mini-courses, held either before a course starts or concurrently with the course at the beginning of the semester, can be very helpful. This is no panacea; many students need the time for difficult concepts and methodologies to sink in, and many don't realize how much they need this extra training. On the other hand, in our experience, logically disadvantaged students can profit from intensive immersion and practice.

- *Instructional software.* Some universities (Carnegie Mellon University, for one) have offered elementary logic instruction through tutoring software. Good textbooks with tutoring packages have been published; see especially (Barwise & Etchemendy 1999). Although software packages can provide useful instruction in the "art of logic," there is as far as we know no instructional software that deals with the much more challenging area of metatheory.

- *Tutoring and labs.* Some universities (for example, the City College of New York) have math labs where students can get free tutoring in remedial math, calculus, and courses like linear algebra and differential equations. The only way students can succeed is through extended practice; environments where drill is customary and pleasant are critical for successful logic instruction, especially for students who have been underexposed to logic. Once the importance of logic in the curriculum is established, it could become possible to argue successfully for logic labs.

  It might be difficult to argue successfully for such a use of university resources on the basis of a need in only one department, or even one school. But developments of this kind might be possible with more systematic cooperation between the various units across the university that are involved in logic instruction: Computer Science, Information Science, Mathematics, Linguistics, and Philosophy. This sort of cooperation is essential if we believe in creating a university climate in which KR instruction can flourish.

- *Written guidelines, examples, and texts.* There are many good logic textbooks, but it is hard to find one that does an adequate job of teaching formalization techniques. (Barwise & Etchemendy 1990) is one of the best recent introductions to formalizing natural language examples, with a range of examples and exercises, and extended discussion of the relation between natural language and logic. The Tarski's World domain used in the tutorial software that accompanies this textbook raises the issue of "micro-formalization"—of how to formalize microworlds in the Tarski's World domain. But the book offers little systematic help in macro-formalization, the art of formalizing extended, non-trivial domains. The most extended textbook discussions of these matters that we know of are (Genesereth & Nilsson 1987, Chapter 2) and the chapters on formalizing specific domains (e.g., Chapters 6 and 7) in (Davis 1990). (Davis 1999) offers a very useful outline of techniques for formalizing commonsense knowledge; a fleshed out outline would be an immense help. Sharing (on the web) some step-by-step approaches to macro-formalization would be very useful; ultimately, we need one or more published handbooks devoted to instruction in formalization techniques.

2.2 LACK OF MOTIVATION

There are two kinds of computer science students: those who want to program and build systems, and those who are interested in the more theoretical aspects of computer science. Students who want to build systems relate to what is immediately practical: programming courses, software engineering, robotics. They are excited by today's hot topics: the internet (and all the dot-com millionaires), e-commerce, computer animation. In some areas, there has been an application-driven shift away from KR methodology: natural language processing courses are more likely to

stress corpus-based and statistical techniques than the KR-based NLP techniques that were popular ten years ago.[2]

Of course, KR is not in the least irrelevant, though it may be seen as irrelevant for application-oriented students who crave immediate gratification. In fact, it is central to many of the most exciting new fields, such as knowledge management and information retrieval. We believe that the fundamental challenge faced by most industrial applications of AI is the problem of how to represent and reason with knowledge. Examples of such industry projects with which one author of this paper (Leora Morgenstern) is familiar include the development of an expert system for benefits inquiry in the medical insurance industry, the automatic configuration of life insurance and banking products; the development of an autonomous agent for searching out and understanding web pages, and diagnostic and helpdesk systems. There is only so far that clever programming techniques will get you. In the final analysis, if you get the knowledge representation part wrong in addressing these problems, the system won't work.

Meeting the Challenge:

This problem is largely a matter of presentation and public relations. Many practitioners of KR care as passionately about the importance of applications as they do about theory, and as KR matures, case studies of its usefulness are becoming more available. In teaching KR, we ought to begin with some of these case studies. Examples can be found in (Swift et al. 1994), (Moore 1995, Chapter 2), (Fikes & Farquhar 1997), (Morgenstern 1998), (Cui et al. 1999), (Brachman et al. 1999), and (Kautz & Selman 1999).

We also ought to put some serious thought into how KR can contribute to popular internet applications like information retrieval, text summarization, and e-commerce. One way of demonstrating KR's relevance would be to examine the inner workings of today's search engines and e-commerce applications, analyze these systems' shortcomings, and find ways in which adding KR methodologies could improve scale and performance. This exercise would be useful not only for teaching KR, but for the health and continued growth of KR as well. This point—the importance of applications for the good of the KR community— was argued in (Morgenstern 1998) for the particular field of nonmonotonic logic and inheritance; the argument needs to be made for KR as a whole.

[2]Compare (Jurafsky & Martin 2000), a textbook that is reasonably comprehensive, with, say, (Gazdar & Mellish 1989).

Theoretically minded computer science students are often attracted by the elegance and self-containment in theory of computation. This elegance is missing in typical introductory KR courses: there is little elegance, for instance, in introductions to first-order theories of planning and the use of frame axioms, where students laboriously grind through axioms to prove that Block A is still on Block B after Block C is moved to Block D. The better-developed areas of KR-related theory have as much elegance as any area of theory.[3] Even if theory can't be the focus of an introduction to KR, theory-minded students need to be made aware of the opportunities and achievements in this area.

2.3 SCARCITY OF TOOLS

Teaching KR could greatly benefit from a set of easy-to-use tools: e.g., tools for standard inheritance (classification and subsumption), tools for inheritance with exceptions, and logic programming tools. Such tools are useful for at least two reasons: First, using such tools allows students to easily see how KR can be used to solve interesting problems. Second, assigning meaningful and useful KR programming projects is almost impossible without tools. If students are required to develop the tools to implement basic KR methodologies from scratch, there will be little time to do the more exciting work that they need to do in order to remain interested in KR as a field.

Meeting the Challenge:

This challenge has already been partly met. Many teachers and research groups have generously devoted their time to making easy-to-use tools available to use: e.g., CMU's repository of KR software at

http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/kr/systems/0.html,

Vladimir Lifschitz's collection of planning software tools at

http://www.cs.utexas.edu/users/vl/teaching/planning.html,

which include, among others, a link to software for Datalog at

http://www.dbai.tuwien.ac.at/proj/dlv,

one to Smodels software at

http://www.tcus.hut.fi/Software/smodels,

and one to materials made available by Eric Sandewall's and Patrick Doherty's group at Linköping at

http://www.ida.liu.se/%7Epatdo/kplabwebsite/software.htm.

[3](Lifschitz 1991), (Kautz & Selman 1989), (Nebel 1990), (Kraus et al. 1990), (Hodges 1994), (Borgida & Patel-Schneider 1994), and (Fagin et al. 1995), are examples. There are, of course, many others.

Many readers will see important omissions in this list. Its partiality indicates another need: for readily available, up-to-date indices of tools that can be useful in teaching KR. Besides obtaining a reasonably complete list of available resources, there is a need for further work in this area. For example, despite the fact that polynomial time algorithms for inheritance with exceptions have been known for over a decade (Horty et al. 1990) and (Stein 1992), there are, as far as we know, no currently available web tools that students can use.

We can fix these problems with some time, effort, and money. Researchers' time and money are always in short supply, but the realization that the investment in time now will pay for itself in the future should help convince many of us to set aside time for this purpose. Money is also always in short supply. We could consider applying for grants from the National Science Foundation or other funding sources to enable the development and dissemination of these tools.

2.4 LACK OF A STANDARDIZED CURRICULUM

There is currently no standard curriculum for Knowledge Representation. This may be partly because there is no standard textbook. (And there may be no standard textbook because the field is so new and fast-moving.) The problem is compounded by the fact that there is so much material to teach, and so little time to do it in. (Unlike courses such as databases or theoretical computer science which often are given over two semesters, with plenty of time for the exploration of various topics, KR, if it is at all offered at a university, is a one-semester course.)

KR teachers have very different ideas of what a KR course should include. Should one concentrate on formalizations in first-order logic, focus on particular domains such as physical and spatial reasoning, integrate the material at the outset with practical applications, include semantic networks or Bayesian networks, include non-standard logics such as nonmonotonic logics, modal logics, and multi-valued logics?

Clearly it is impossible to cover all of these topics (with any degree of thoroughness) in one course, but some agreement would be helpful on what subset of these topics is essential for grounding students in KR.

Meeting the Challenge:

A standardized curriculum would help to develop tools and teaching materials, and a consensus on what belongs in the curriculum would help the KR community present a united front on the subjects they consider most representative and important. We address these issues in Section 3.

2.5 TEACHING KR VS. USING KR IN THE FIELD

It is interesting to note that many of the problems that arise in teaching KR at a university often occur in a strengthened form when practitioners use KR to solve problems in industry. Such an effort almost always involves a partnership between the researhers and developers of the KR solution, on the one hand, and the users of the solution, on the other hand.

For example, in developing an expert system for benefits inquiry in a medical insurance corporation, one of the authors (Leora Morgenstern) used as the central structure a semantic network, consisting mostly of a taxonomy of medical services, augmented by business rules. The population of the semantic network was a joint effort between her and a small group of employees in the insurance corporation. These employees had to receive a crash course in semantic networks. They had to be taught how to identify the core concepts in the domain (in this case, medical procedures, conditions, and settings), how to enumerate the nodes in the network, and how to organize the nodes. Once the system was developed and in use, the policy modifiers — those insurance company employees who change insurance policy rules — had to learn how to change the network by adding, deleting, and modifying nodes and rules. Teaching these employees how to use semantic networks was quite similar to teaching semantic networks in a classroom (although users of commercial systems are already quite motivated).

We are not suggesting that developing KR for commercial applications entails teaching a full-fledged KR course in the field. However, we do wish to stress the connection between the two activities. We believe that many of the general methodologies that we develop to bring KR students up to speed can also be used when developing methods to train commercial users of KR-based systems. Conversely, if we are poorly prepared to teach KR to unsophisticated students, it will be that much harder to effectively use KR in the commercial world.

The realization that teaching KR to university students is closely related to training customers to use commercial KR systems should serve as added motivation to solve the problems that we face.

3 COURSE ORGANIZATION: SYLLABUS, MATERIALS, AND SOURCES.

3.1 THE SYLLABUS

As we mentioned, there is currently no consensus on what belongs in a KR course. We make some suggestions in this section, in the hope that this will at least serve to stimulate discussion on the topic within the KR community.

We suggest that a KR course include at the minimum a review and discussion of first-order logic; an analysis of why first-order logic is not in itself sufficient for the KR enterprise, an example of an extension of first-order logic (e.g., modal logics or logics augmented by quotation or a nonmonotonic logic), semantic networks, inheritance with and without exceptions, temporal logic and planning, and a discussion of how KR systems have been used in real-world applications. Other topics which may be included, time permitting, include a more detailed discussion of modal logic, the study of some particular domains of commonsense reasoning such as spatial or physical reasoning, logic programming, specific nonmonotonic formalisms, explanation, and Bayesian nets.

Below is a possible syllabus for a KR course. We assume a fourteen-week semester with classes meeting twice a week, giving twenty-eight classes. Some topics are listed below, with the assumption that they will be gone through in this order with anywhere from 2-4 sessions on each topic.

1. Basic principles of KR and motivation. Case study of systems using KR, with demos. Case study of systems not using KR (like today's search engines); discussion of how KR methodologies could be used to improve them. Use these initial discussions for ideas for projects later on.

2. Basic first-order logic. Introduction/review, practice and more practice. Very brief overview of why first-order logic is not enough to represent general knowledge, and why modal logic and nonmonotonic logics will be needed.

3. Logic Programming methodology.

4. Semantic networks and examples of applications (e.g., from natural language processing).

5. Inheritance networks with exceptions, brief mention to nonmonotonic logics.

6. Temporal logic. The situation calculus, perhaps the event calculus, examples of planning.

7. Basics of modal logic. Epistemic logics; examples of applications from distributed computing (Fagin et al. 1995) ; Deontic logics.

8. Basics of nonmonotonic logics. Circumscription, default logic, autoepistemic logic. Nonmonotonic semantics (negation as failure) for logic programming.

9. Comparison with other formal methodologies such as probabilistic reasoning. Bayesian networks.

10. Using above techniques for applications. Detailed examples.

3.2 MATERIALS, TOOLS, AND SOFTWARE

A successful KR course needs to be firmly grounded in both theory and practice. As such, it would be helpful to provide students with both written materials and software tools.

Written Materials

As we have already said, there is a dearth of KR textbooks. Certainly no existing textbook covers, for example, the syllabus suggested in the previous section. Still, the available texts are useful as a starting point. They should be augmented, of course, with readings. The fifteen-year-old collection (Brachman & Levesque 1985), while of course out of date, is nevertheless important because it contains so many classic papers. In addition, there are many excellent recent papers on specific topics, and general encyclopedia articles that can give students good overviews (e.g., (Davis 2001a), (Hayes 1999) , and the older (Barr & Davidson 1981). As for newer collections, (Brachman et al. 1992) is helpful, as well, of course, as the KR Conference Proceedings from 1989 onwards. Teachers can of course compile lists of important readings covering topics not addressed in existing texts; such lists may consist both of hard-copy papers (distributed in class or put on reserve in the library) or links to on-line papers. It would be very useful to have a central website where such lists could be maintained; this would be an important step towards achieving consensus on a curriculum.

In addition to textbooks and papers, the authors have found that students often find slide presentations very helpful. We are not entirely comfortable with encouraging a reliance on quick summaries and easy sound bites; nevertheless, it is hard to argue with anything

that can help students plow their way through difficult material. There is a wealth of material on the web, including slides that have been used for tutorials at conferences and slides that have been used in individual classes. Examples include a collection of tutorials for description logics

http://www.cs.man.uk/~franconi/dl/course ;

sets of slides accompanying the introductory textbook (Poole et al. 1998)

http://www.cs.ubc.ca/spider/poole/ci/lectures/lectures.html),

and the websites of the authors' courses. A central website with links to these materials would be very useful.

Software Tools

Ideally, the software tools used for a KR course should meet the following requirements:

- they should be quick to learn and easy to install and use;
- they should be platform independent, or at least, be available on a variety of platforms and not require installation of too much accessory software;
- they should serve as the basis of useful and interesting applications; that is, students using the software ought to be able to develop non-trivial KR applications with relative ease;
- they should be neutral with respect to ongoing ideological wars in the KR community.

We don't know of an existing substantial library of software tools meeting the above requirements; we hope that the creation of such a library will occur sometime in the future. Such a library could include general theorem-proving tools including logic programming software and nonmonotonic theorem provers, semantic network tools, tools for inheritance with exceptions, Bayesian reasoning tools, and tools for temporal reasoning and planning.

One of the authors (Rich Thomason) has found the CLASSIC system to be an valuable tool in teaching KR. The clean design and excellent documentation of this system make it especially appropriate for teaching students; at the same time, it provides a suitable framework for serious representation projects. Often, these projects can be integrated into larger systems on which students are working. An online tutorial for CLASSIC is available at http://www.eecs.umich.edu/ rthomaso/kr/classic-tutorial.html

Of course, the need for a central website with pointers to useful KR software is just as great as (if not greater than) the need for an index of written materials.

4 FUTURE PLANS

We believe that a centralized effort is necessary in order to strengthen the role of KR in the AI and CS curriculum. Specifically, we believe that it may be helpful for the KR organization to become more active in supporting and promoting the teaching of KR. The following possibilities have occurred to us; others may emerge from discussion at KR2000.

1. Establishing a website devoted to teaching KR. Such a website could include the libraries for written materials and software tools discussed in the previous section.
2. Gathering information concerning perceived problems and solutions. It might be useful to conduct some sort of survey of those teaching KR, to find out the problems that they have encountered and how they have tried to address these problems. The results of this survey could be shared with the KR community. We have doubtless faced many of the same problems, and sharing solutions could save time and energy.
3. Preparing a position statement on the role of KR in the Computer Science curriculum, and sharing it with the KR community. Such a position paper could be used—in whole or in part—by KR faculty who need to strengthen the role of KR in their departments, and could also be useful in writing grant applications requesting funding for the development of KR software tools.

All of this will involve a good deal of work. Of course, we can't realistically expect more to be done than individuals can manage to do on an unpaid basis. Still, historically, much academic progress has been made possible by volunteer labor. The usual machinery that has facilitated and leveraged this labor—the formation of a small committee or task force in charge of these issues, a short-term mandate to construct a website of the sort discussed above (along with a way to count it as a publication)—should stand us in good stead here. We expect that the work involved will strengthen the position of KR in the AI and CS communities.

References

[1] Avron Barr and James E. Davidson. Representation of knowledge, vol. 1. In Avron Barr and

Edward A. Feigenbaum, editors, *The Handbook of AI*, pages 141–222. HeurisTech Press, Stanford, California, 1981. Co-authors: Robert Filman, Douglas Appelt, Anne Gardiner, and James Bennett.

[2] Jon Barwise and John Etchemendy. *Language, Proof, and Logic*. CSLI Publications, Stanford, California, 1999.

[3] Alex Borgida and Peter Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[4] Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter, editors. *Knowledge Representation*. The MIT Press, Cambridge, Massachusetts, 1992.

[5] Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lori A. Resnik. "reducing" CLASSIC to practice: Knowledge representation theory meets reality. *Artificial Intelligence*, 114(1–2):203–237, 1999.

[6] Ronald Branchman and Hector Levesque, editors. *Readings in Knowledge Representation*. Morgan Kaufmann, San Francisco, 1985.

[7] Baoqui Cui, Terrance Swift, and David S. Warren. A case study in using preference logic grammars for knowledge representation. In Michael Gelfond, Nicola Leone, and Gerald Pfeifer, editors, *Lecture Notes in Artificial Intelligence 1730: Logic Programming and Nonmonotonic Reasoning*, pages 206–220, Berlin, 1999. Springer-Verlag.

[8] Ernest Davis. *Representations of Commonsense Knowledge*. Morgan Kaufmann, San Francisco, 1990.

[9] Ernest Davis. Guide to axiomatizing domains in first-order logic. *Electronic Newsletter on Reasoning about Actions and Change*, 99002, 1999. http://www.etaij.org/rac/.

[10] Ernest Davis. Knowledge representation. In Neil J. Smelser and Paul B. Baltes, editors, *The International Encyclopedia of the Social and Behavioral Sciences*. Elsevier, Amsterdam, 2001.

[11] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. The MIT Press, Cambridge, Massachusetts, 1995.

[12] Richard Fikes and Adam Farquhar. Large-scale repositories of highly expressive reusable knowledge. Unpublished manuscript, Knowledge Systems Laboratory, Stanford University, 1997.

[13] Gerald Gazdar and Chris Mellish. *Natural Language Processing in Prolog: An Introduction to Computational Linguistics*. Addison-Wesley Publishing Co., Reading, Massachusetts, 1989.

[14] Michael Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, San Mateo, California, 1987.

[15] Patrick Hayes. Knowledge representation. In Robert A. Wilson and Frank C. Keil, editors, *MIT Encyclopedia of the Cognitive Sciences*. The MIT Press, Cambridge, Maassachusetts, 1999.

[16] Wilfrid Hodges. Logical features of Horn clauses. In Dov Gabbay, Christopher Hogger, and J.A. Robinson, editors, *The Handbook of Logic in Artificial Intelligence and Logic Programming, Volume I*, pages 449–503. Oxford University Press, Oxford, 1994.

[17] John F. Horty, Richmond Thomason, and David Touretzky. A skeptical theory of inheritance in nonmonotonic semantic networks. *Artificial Intelligence*, 42:311–349, 1990.

[18] Daniel Jurafsky and J. Martin. *Speech and Language Processing*. Prentice Hall, Englewood Cliffs, New Jersey, 2000.

[19] Henry Kautz and Bart Selman. Hard problems for simple default logics. In Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter, editors, *KR'89: Principles of Knowledge Representation and Reasoning*, pages 189–197. Morgan Kaufmann, San Mateo, California, 1989.

[20] Henry Kautz and Bart Selman. Unifying SAT-based and graph-based planning. In Thomas Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 318–325, San Francisco, 1999. Morgan Kaufmann.

[21] Robert A. Kowalski. English as a logic programming language. *New Generation Computing*, 8(2):91–93, 1990.

[22] Robert A. Kowalski. An undergrauate degree in practical reasoning. *Journal of Logic and Computation*, 3(3):227–229, 1993.

[23] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 14(1):167–207, 1990.

[24] Vladimir Lifschitz. Towards a metatheory of action. In James Allen, Richard Fikes, and Erik Sandewall, editors, *KR'91: Principles of Knowledge Representation and Reasoning*, pages 376–386. Morgan Kaufmann, San Mateo, California, 1991.

[25] John McCarthy. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pages 75–91, London, 1959. Her Majesty's Stationary Office.

[26] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[27] Marvin Minsky. *Semantic Information Processing.* The MIT Press, Cambridge, Massachusetts, 1969.

[28] Johanna Moore. *Participating in Explanatory Dialogues.* The MIT Press, 1995.

[29] Leora Morgenstern. Inheritance comes of age: Applying nonmonotonic techniques to problems in industry. *Artificial Intelligence*, 103(1–2):237–271, 1998.

[30] Bernhard Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43(2):235–249, 1990.

[31] Nils J. Nilsson. Eye on the prize. Available at www.robotics.stanford.edu/ nilsson., 1995.

[32] David Poole, Alan Mackworth, and Randy Goebel. *Computational Intelligence: A Logical Approach.* Oxford University Press, New York, 1998.

[33] Erik Sandewall. On the methodology of research in knowledge representation and common-sense reasoning. In Jack Minker, editor, *Logic-Based Artificial Intelligence.* Kluwer Academic Publishers, Dordrecht, 2000. Forthcoming.

[34] John F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations.* Thomson Learning, Stamford, Connecticut, 1999.

[35] Lynn Andrea Stein. Resolving ambiguity in nonmonotonic inheritance hierarchies. *Artificial Intelligence*, 55(2–3):259–310, 1992.

[36] T. Swift, C. Henderson, R. Holberger, J. Murphey, and E. Neham. Cctis: An expert transaction processing system. In *Proceedings of the Sixth Conference on Industrial Applications of Artificial Intelligence*, pages 131–140, 1994.