THE BASICS OF KNOWLEDGE REPRESENTATION AND REASONING

Ronald J. Brachman

Ronald J. Brachman *is head of the Artificial Intelligence Principles Research Department at AT&T Bell Laboratories in Murray Hill, New Jersey. Before joining AT&T in 1985, he led the design and implementation of two well-known knowledge-representation systems, KL-ONE and KRYPTON. Mr. Brachman received a B.S. in electrical engineering from Princeton University and master's and Ph.D. degrees in applied mathematics from Harvard University.*

A widely recognized goal of artificial intelligence (AI) is the creation of artifacts that can emulate humans in their ability to reason symbolically, as exemplified in typical AI domains such as planning, natural language understanding, diagnosis, and tutoring. Currently most of this work is predicated on a belief that intelligent systems can be constructed from explicit, declarative knowledge bases, which in turn are operated on by general, formal reasoning mechanisms. This fundamental hypothesis of AI means that knowledge representation and reasoning—the study of formal ways of extracting information from symbolically represented knowledge—is of central importance to the field. In this article, we review some of the basics of this important research area and briefly survey the kinds of techniques typically used for representation in AI programs. We also consider some important current research directions.

Introduction

Almost all current attempts at artificial intelligence systems have distinct knowledge bases (KBs), which are explicit, declarative statements of the system's knowledge about its domain. For instance, the typical expert system has its domain knowledge (of, say, medicine) encoded as a set of rules of thumb (for example, relating disease symptoms to their probable causes). Knowledge in a KB is encoded in some formal language, and is operated on by a domain-independent interpreter. This general-purpose reasoning mechanism looks at the KB and the facts about the current situation (e.g., the patient's current symptoms) and draws inferences, leading it, one hopes, to some important conclusion or solution to a posed problem (e.g., a diagnosis or course of therapy).

This simple picture of a set of explicit domain facts and a general "inference engine" is by now a cliché, and is by itself unrevealing.

However, this simplistic view is really a reflection of a more significant and fundamental hypothesis about intelligence—one that is filled with substantial implications for how work in the field should proceed and be judged. Brian Smith calls this the *knowledge representation hypothesis*:[1]

"Any mechanically embodied intelligent process will be comprised of structural ingredients that (a) we as external observers naturally take to represent a propositional account of the knowledge that the overall process exhibits, and (b) independent of such external semantical attribution, play a formal but causal and essential role in engendering the behaviour that manifests that knowledge."

Smith's account highlights two crucial facets of intelligence:

- In order for the symbolic structures of a system to represent knowledge, it must be possible to interpret them propositionally—as expressions in some language that have truth values. In other words, these structures must be akin to sentences, which can be either true or false. In a sense, we should be able to point to one of them and say what the world would have to be like for it to be true. This means that the symbols used in forming the sentences must subscribe to a consistent interpretation.
- It must be the presence of these structures that causes the system to behave in the way that it does. This is crucial, since sentences that were like comments in a program would be of no consequence to intelligent behavior.

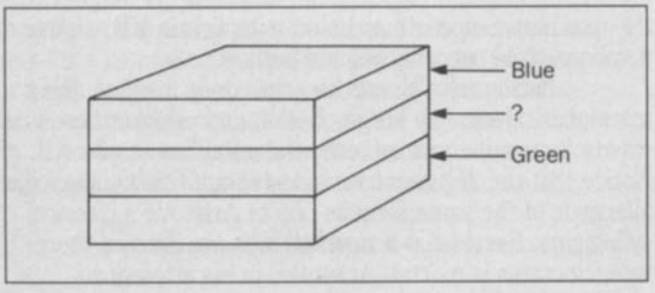
While the claim that intelligent behavior can arise out of the computational manipulation of propositional symbolic structures is certainly open to argument, AI has not yet developed a serious alternative to this working hypothesis. At the moment, this stance is responsible for the vast majority of work in the field; one could even say that "knowledge-based systems"—virtually synonymous with "AI systems"—are exactly those systems that satisfy the knowledge representation hypothesis by design. Thus it is crucial to understand the consequences of this view, and to use it as a context for judging the adequacy of work in the field.

It should be pointed out that the view of the possibility of mechanized intelligence is not new. It seems to have originated with Leibniz' (1646-1716) dream of a calculus of ideas, wherein truths could be determined by manipulating an "alphabet of thought" (*characteristica universalis*) in some combinatory way, much akin to the way numerical expressions are manipulated in Newton's calculus. The crucial thing about such manipulation is that it would be required to preserve truth, similar to the way that numerical manipulation preserves value.

In any case, the study of interpreted symbolic structures and their formal manipulation in a truth-respecting way is now the cornerstone of AI. While Leibniz was not around to see it, the invention of mathematical logic by Frege at the end of the 19th century provided the crucial technical ingredient for the development of mechanized reasoning. The work of Frege and his successors gives AI a solid technical foundation on which to build reasoning systems.

In AI, studies of the application of logic to reasoning have generally been pursued in the area known somewhat loosely as *knowledge representation* (KR), or *knowledge representation and reasoning*. In general terms, this subfield is concerned with the forms in which knowledge can be expressed, the meanings of such forms, and the complexity of reasoning with them.

Our interest here is with the basic goals and techniques of KR, and in the following sections we consider the logical basis of representation and reasoning and some of the implications that it has for work in the field. We survey the most commonly used representational styles, and conclude with a brief account of some currently important research issues. Along the way, we will mention some relevant efforts here at AT&T. We will not have time to discuss directly any applications of KR technology, but given the ubiquity of the KR point of view, it should be easy to spot the crucial role played by representation in almost all of the other articles in this special issue.



Panel 1. Entailment

Finding the truth implicit in a set of statements about the world is not always as straightforward as putting one and one together—nor is it always a matter of seeing what is entailed by two or more statements. A single proposition may have important entailments that need to be realized by a knowledge-based system. This is because even a single statement about the world refers to a situation (or set of situations), and simply because that situation exists, other facts must hold as well.

For example, consider a simple situation where there are three colored blocks in a tower, resting on the floor. Let's say that I make the statement,

"A blue block is sitting on another block, which is sitting on a green block."

This sentence asserts that the situation in the accompanying figure holds. Because this is the situation represented by the sentence, it is also the case that

"A blue block is on a nonblue block."

This is simply true of the described world, even if it is not obvious by looking. [To determine that this second sentence is indeed entailed by the first, consider the two cases of the middle block being (a) blue or (b) not blue. Since there are no other possibilities, and in each case there is a blue block on a nonblue one, the second sentence follows from the first.]

Entailment is about what's true in a described set of conditions. Once we describe a situation by asserting a proposition, then there are other propositions that are simply forced to be true, just by the nature of the situation.

Foundations of Computational Inference

Throughout its history, KR work has been primarily concerned with particular technical mechanisms for expressing different kinds of knowledge. A great deal of effort has gone into the study and development of various notations and formalisms, with rather less attention paid to logical foundations and inferential consequences of the notations. However, given that this work owes its very existence to the knowledge representation hypothesis, it is important to consider the more basic issues before we can discuss notations and implementations.

Logical Consequence. Imagine that we were building a knowledge-based system to assist in designing a course of medical treatment. Among the things we might want to include in the system's knowledge would be a description of which patients were known to be allergic to which medications. At first, a simple database would seem to do: Ralph is allergic to sulfa, Trixie is allergic to penicillin, Alice is allergic to tetracycline, etc. In this case, we could determine by a simple retrieval operation whether or not to use a certain drug in the therapy.

However, drugs are related to one another, and shared pharmacologic properties can predict similar reactions. We might know, for example, that all patients allergic to penicillin are also allergic to ampicillin. We would certainly want the system's knowledge base to contain such an important, universally true fact. Now, if we were to consider administering ampicillin to Trixie, what

should our system recommend? We would expect the system to realize that the specific fact about Trixie's allergy and the general fact about patients and the two drugs together determine that Trixie will also be allergic to ampicillin. The system would be incompetent if it did not warn against using the drug.

This fact cannot be obtained by a simple retrieval operation. We have not included in our KB any explicit fact about Trixie and ampicillin. Further, it does not require knowledge of medicine to see that the generalization about allergies and the simple fact of Trixie's allergy to penicillin together imply the third fact. It is simply the case that if the world is such that someone is allergic to drug A, and all who are allergic to drug A are allergic to drug B, then the world must be such that the person in question is allergic to drug B. The truth of that conclusion is implicit in the truth of the two premises. In more formal terms, we would say that the conclusion is *entailed* by the premises. (See Panel 1.)

The trick is to get our therapy system to realize this relationship. While the truth of the conclusion is dependent only on the way the world works and a straightforward notion of truth, it will take some work to get a knowledge base to be able to see the logical consequences (entailments) of what is explicitly stated. For one thing, intuitions about the meanings of English words, which are available to us and allow us to make inferences without conscious effort, are not magically present in the machine. In any case, it should be clear that a knowledge-based system must be concerned not so much with what is explicitly stated in its KB, but instead, with what those explicit structures taken together tell it about its domain.

Computing Entailments: Inference. Our primary need is for a knowledge-based system to be able to realize all of the things that the sentences in its KB determine to be true about the world. In other words, when asked a question, α, the system needs to determine whether or not KB $\models \alpha$ ("$\models$" means "entails"). This is a question about the way the world described by the KB is, but our machine cannot just look to see the truth of α—it must have some mechanical (algorithmic) way of computing the answer to the question. Since α may be only implicit in KB, simple database-style retrieval will not suffice.

Fortunately, thanks to some deep insights about mathematical logic by Frege, Gödel, and others, there is a way to determine mechanically the entailments of a KB. Notice that the argument we made about Trixie's ampicillin allergy is of the same form as one of Aristotle's classical syllogisms: Socrates is a man; all men are mortal; therefore, Socrates is mortal. Aristotle, in his attempt to characterize the valid forms of argumentation, determined that a conclusion of a certain form can be seen to follow from premises of a certain form (thus the origin of the term "formal logic"). What is required is consistent behavior on the part of certain special words or symbols, namely "all" and "is" and "are." The particular conclusions and premises are not of concern as much as the forms used in a valid derivation.

The derivation of a valid conclusion is not dependent on the meaning of any of the "nonlogical" symbols, like "Socrates," "man," and "mortal." Rather, what is taken into account is only the form of the two premises, along with a *rule of inference* that states that, if it is given that $\forall\, x\, p(x) \supset q(x)$ ("for any x, if x has property p, then it has property q"), and it is given that $p(a)$ for some particular a ("a has property p"), then it is acceptable to conclude that $q(a)$ holds. We can say that the conclusion is *derivable* from the given premises, and write

$$\forall\, x\, p(x) \supset q(x),\ \ p(a) \vdash q(a)$$

(the symbol "$\vdash$" means "derives"). The notion of a formal proof is then specified as any sequence of formulas such that each member of the sequence is given or is derivable from previous members by a rule of inference. The given formulas are called *axioms*; the derived formulas are called *theorems*.

Since proofs are just a matter of form, and not content, it would seem that they are just the kind of symbol manipulation that computers are good at. If we could

show this for certain, and could tie the notion of derivation directly to the notion of entailment, then we would have a way for a computer to mechanically produce facts implicit in a KB. Indeed, there are two crucial relationships between entailments and derivations that close the loop for us: soundness and completeness theorems for predicate logic tell us that for a set of sentences, KB,

$$KB \models \alpha \text{ if and only if } KB \vdash \alpha$$

That is, if something is implicitly true in a KB, there exists a proof for it (and vice versa). Further, it is indeed possible to mechanize proof procedures for standard logics, and it is on this possibility of "automatic theorem proving" that the entire enterprise of knowledge representation and reasoning rests. If the logical language we use to express our facts is appropriate, then we can indeed write a computer program that directly manipulates symbols like "Trixie," "ampicillin," "is," and "all," and determines that we should avoid prescribing the drug for our patient.

To recapitulate, it is central to the goals of artificial intelligence that knowledge-based systems be able to determine truths about the world implied by the contents of their knowledge bases—to go beyond the explicitly stated facts. A deep correspondence between this "semantic" relation of entailment and the "syntactic" relation of derivability (inference) suggests that it is possible to implement systems that can indeed infer truths about the world by formal manipulation of symbols. It is the notions of logic and theorem-proving that make the knowledge representation hypothesis coherent, if not plausible.

Before moving on, we should note that our investigation of particular KR formalisms will appear to take us far out of the realm of standard logics (like that of first-order predicate calculus) and theorem-proving. KR systems come in many styles and forms, with a veritable Babel of notations, from graphs to heuristic rules of thumb. Most have been developed by looking only at notational syntax, with utility generally assessed by appeal to intuitive mappings onto isolated natural language examples. But despite the apparent diversity and lack of resemblance to first-order predicate calculus, it is crucial for us to remember that knowledge representation all comes down to logic and theorem-proving. We should always keep in mind that meaning is primary; despite the appeal of intuitive data structures, interpretation and entailment should always be in the driver's seat. In order to be taken seriously, a KR language must come with an interpretation specified. This gives an independent measure of correctness of the code and tells the user what inferences to expect the system to be capable of drawing. Otherwise, the supposed KR system is simply a means for manipulating meaningless data structures, and will not connect to the primary goal of computing entailments.

A Fundamental Tradeoff

The notion of derivability, so fundamental to the mechanical proof of theorems, depends only on the existence of certain sequences of formulas, related by rules of inference. It does not depend on whether a particular derivation can be found in a reasonable amount of time.

Unfortunately, if we take standard first-order logic (FOL), which uses quantifiers like "all" and operators like "and" and "not," as well as predicate and function symbols, deciding whether a sentence is a theorem is in general unsolvable.[2] That means that our therapy assistant, if given knowledge of the right form (i.e., that uses the full expressiveness of FOL), may never return with an answer to a question. This is clearly unacceptable in most contexts. Even if we eliminate the quantifiers from the KB, computation of theoremhood can be intolerably expensive (the problem is "NP-hard" for propositional logic, and therefore believed to have at best an exponential solution).[3] These are worst-case analyses, but since we cannot in advance predict how good a case we have, or what an average case might look like, we must always be prepared to fall off the computational deep end.

This bodes rather poorly for the whole notion of mechanized inference; at the very least it demands the direct attention of those in the business of building AI sys-

tems. The trickiness here is that the complexity barrier is not a matter that can be taken care of by clever algorithm design or optimization—it is inherent in the problem of reasoning itself. Nor is it the fault of the specific language of standard first-order logic—any language as precise as FOL that allows the same kind of distinctions to be made is inherently just as intractable (any language sufficient to axiomatize arithmetic will have this problem, for example). The problem is that FOL allows us to leave many things unspecified—a desirable property in building knowledge bases—but this incompleteness multiplies the number of cases to be considered, sometimes infinitely.

This leaves us with an imperative for research in knowledge representation: since the problem is insurmountable in the general case, are there ways to avoid it or live with it in special cases? There are several alternatives: we can try to push back the computational barriers by developing special-purpose hardware or algorithms, and hope that we can sufficiently cover the commonly occurring cases; we can relax correctness, and be prepared to accept incorrect answers from our reasoning system; or we can use logics simpler than FOL, either specially designed for use in limited cases, or with weaker notions of entailment.

Indeed, experience shows us that we have some hope. Systems have been constructed that are far weaker than full FOL, but have proven useful in limited practical applications. But the fact is that in the KR business we must be prepared to bite the intractability bullet—the difficulty of calculating entailments in sufficiently expressive languages is an ever-present obstacle. As Hector Levesque puts it, " . . . we can understand much of KR research as attempts to reconcile simultaneously the demands of logical coherence and computational tractability"[4] (for a more thorough treatment of the tradeoff between complexity and expressiveness as applied to KR, see Reference 5).

Some Common Knowledge Representation Schemes

The common techniques of knowledge representation that we will now discuss can be understood as attempts to deal with only limited kinds of knowledge in exchange for reasonable complexity behavior. It should be pointed out, however, that this is rarely the view taken by those responsible for the invention and development of these commonly used representational formalisms (except for predicate logic). Most attention has been paid to matching constructs in the representation directly with phenomena in the domain, and to considerations of implementation. This has indeed led to a number of undeniably useful tools for building knowledge bases and inferring things from them. However, since so little attention has been paid directly to the semantics of the languages, there have been occasions where systems have not met users' expectations about their behavior, not to mention cases where systems have been found to have inconsistent or meaningless behavior (the interested reader should consult References 6 and 7 for discussions of particular cases).

In the KR work being pursued at AT&T Bell Laboratories, we have paid close attention to the formal semantics of our representation logics, as well as to the computational complexity of our inference problems (see below). Also, recent work in the field has begun to be infused with a sense of respect for both semantic and complexity issues.

Using Predicate Logic Directly. In a sense, artificial intelligence as a whole got its start when John McCarthy suggested in 1956 that interesting commonsense problems could be solved by manipulation of expressions in predicate calculus.[8] Since that time, one of the important strains of KR research has used classical forms of first-order logic directly in programs to represent knowledge. This type of representation has been especially useful in formal approaches to planning, where predicates are used to represent situations and goals, such as *at(I,airport)* or *on(blockA,blockB)*, and deduction rules are used to correspond to actions.

The goal of a planning system is to find a sequence of executable actions that will transform the initial state to a state that satisfies the goal. This can be

thought of directly as a theorem-proving task, with the goal state being the theorem we wish to prove and the initial state an axiom (or set of axioms); the rules of inference correspond to actions that transform one state into another. (This is an oversimplification. See the article by Kautz and Pednault[9] in this issue for a discussion of the complexities involved in real planning.)

Natural language processing programs (see the paper by Hirschberg et al.[10] in this issue) have also made extensive use of formal logic representations, primarily because the nuances of interactions in languages like English demand great expressive power of whatever language is used to represent word, sentence, and discourse meanings. In many cases, there seems to be a somewhat direct map from English (especially expressions with quantifiers) onto predicate calculus expressions. For example, we might represent a sentence like "no woman likes a man who does not like all vegetarians" as

$$\neg\, \exists\, x \exists\, y \exists\, z \text{ Woman } (x) \wedge \text{ Man } (y) \wedge \text{ Vegetarian } (z) \wedge \neg \text{ Likes } (y,z) \quad \text{Likes } (x,y)$$

Or, we may take

$$\neg\, \forall\, x \text{ Glitters } (x) \supset \text{ Gold } (x)$$

to mean "not everything that glitters is gold." While it is easy to get started this way, there are many subtleties here, involving the correct interpretation of natural language quantifiers, "opaque" belief contexts, pronoun resolution, etc., and it is not clear that standard logics are sufficient or appropriate for the full range of needs of natural language programs.

The great appeal of using formal predicate logic directly, of course, is the existence of the standard theorem-proving procedures that respect meaning, as mentioned above. The primary key to the implementation of mechanical deduction systems has been Robinson's *resolution* rule of inference,[11] which yields a simple straightforward algorithm for proving theorems. Since 1965, there has arisen a substantial automatic theorem-proving (ATP) community, with a raft of implemented theorem-provers, as well as numerous refinements to resolution that attempt to speed up the deduction process. A number of special-purpose theorem-provers, built more for AI applications than for general-purpose ATP, have also been implemented.

The advantage of using formal logic as an implementation medium for knowledge representation systems—its generality and the existence of general deduction procedures—is also its Achilles' heel. As noted above, computational complexity can be deadly (theorem-proving for full first-order logic is in general undecidable). While many of the refinements have helped to streamline resolution systems, they can never completely overturn the inherent intractability. As a result, the ultimate role of general-purpose ATP in AI is still quite cloudy. ATP programs have been of help to researchers attempting to solve open problems in mathematics (which may take months, and require significant user intervention in proofs). But the mode of operation in AI programs attempting to deal with everyday commonsense problems is so different that ultimately, general theorem-provers may be of little use in everyday domains.

In any case, formal logic is certainly an important tool for modeling and understanding the consequences of what is in a knowledge base,[12] and despite its potential problems, its direct use in AI programs has been important. For a comprehensive treatment of logic in AI, the reader is referred to a new textbook by Genesereth and Nilsson.[13]

Production Systems. Since the advent and widespread exposure of expert systems, production rules have become one of the most popular means of representing knowledge in AI systems. In certain tasks—especially diagnosis and configuration—it appears that what an expert knows about a domain can very conveniently be encoded as a set of "if—then" associations. For example, a rule like this might be useful in the context of an expert system for processor design (see the article by Kowalski[14]

in this issue):

IF: the most current active context is bus allocation
and there is a link from a module to a bus
and there is a link from the bus to a multiplexer
and there is a link from the module to the multiplexer
THEN: check to see if the bus is idle during the control step needed to multiplex the values

Similarly, in a medical context, we might see a rule whose antecedent (the "if" part) recognizes certain symptoms and whose consequent (the "then" part) increases confidence in a disease hypothesis.

A production system generally has three parts:

- A rule base, which holds the complete set of rules that constitute the generic knowledge of the system; sometimes these rule sets have substructure.
- A working memory, which contains elements that represent the current particular situation; these elements usually represent the situation in a fairly simple fashion, using attribute/value pairs.
- A rule interpreter, which applies the rules in some cyclical fashion, changing working memory according to the consequents of rules found applicable to the current state.

The interpretation cycle proceeds by (in essence) matching the antecedents of all rules against all elements in working memory, thereby determining a subset of rules that are currently applicable (the conflict set). Some or all of the rules in the conflict set are activated or "fired," depending on the method of conflict resolution. These activations change working memory according to the consequents of the fired rules, and then the cycle repeats.

Somewhat in contrast to predicate logic, production systems typically force users into thinking about issues relating to the control of the inference process. In order to write a successful production system, one must choose carefully the strategy for conflict resolution (use the most specific rule? use the rule that matches the most recently changed working memory element? use the least recently used rule?). It is also a good idea to segregate a task into subtasks, to keep rules from getting in each other's way. This is the role played by "the most current active context" in the above example.

Production systems are clearly useful for some tasks, although it is awkward to try to represent generalization hierarchies (see the next section) and other kinds of facts in a strict "if—then" format. Typically they are very weak in expressive power, at least when compared to first-order logic, so they on the one hand will be unable to represent certain kinds of facts, but on the other, will run predictably quickly. Technically, they seem to be equivalent to the Horn subset of propositional logic; thus, deciding entailment can be shown to be calculable in linear time.

There are some who believe that production systems are a viable model of virtually all human intelligence, and have developed elaborate frameworks that allow "meta"-rules to help guide the search for an answer to a complicated problem. However, for the most part, the notion of a rule-based representation of knowledge has led to the development of some very useful tools for building systems of a fairly constrained nature. For details on one of them—C5, a version of OPS5—see the article by Vesonder in this issue.[15]

Semantic Networks. Another popular representation technique emphasizes the connectivity of one's knowledge, in analogy to the construction of a dictionary. Each word is defined in terms of others, creating a linked structure that interrelates virtually everything known. This suggests a gigantic network, with word meanings tangled together in an associative fashion. This is the inspiration for the semantic network, a linked data structure that directly reflects the massive connectivity of knowledge.

Semantic nets have evolved into a large family of related, but often quite different, systems. Despite differences in detail, they generally share a number of features: nodes in the graphs represent individual entities in the domain, or classes of entities; links between nodes represent binary relationships between the things denoted by the nodes. There is usually a special link that designates

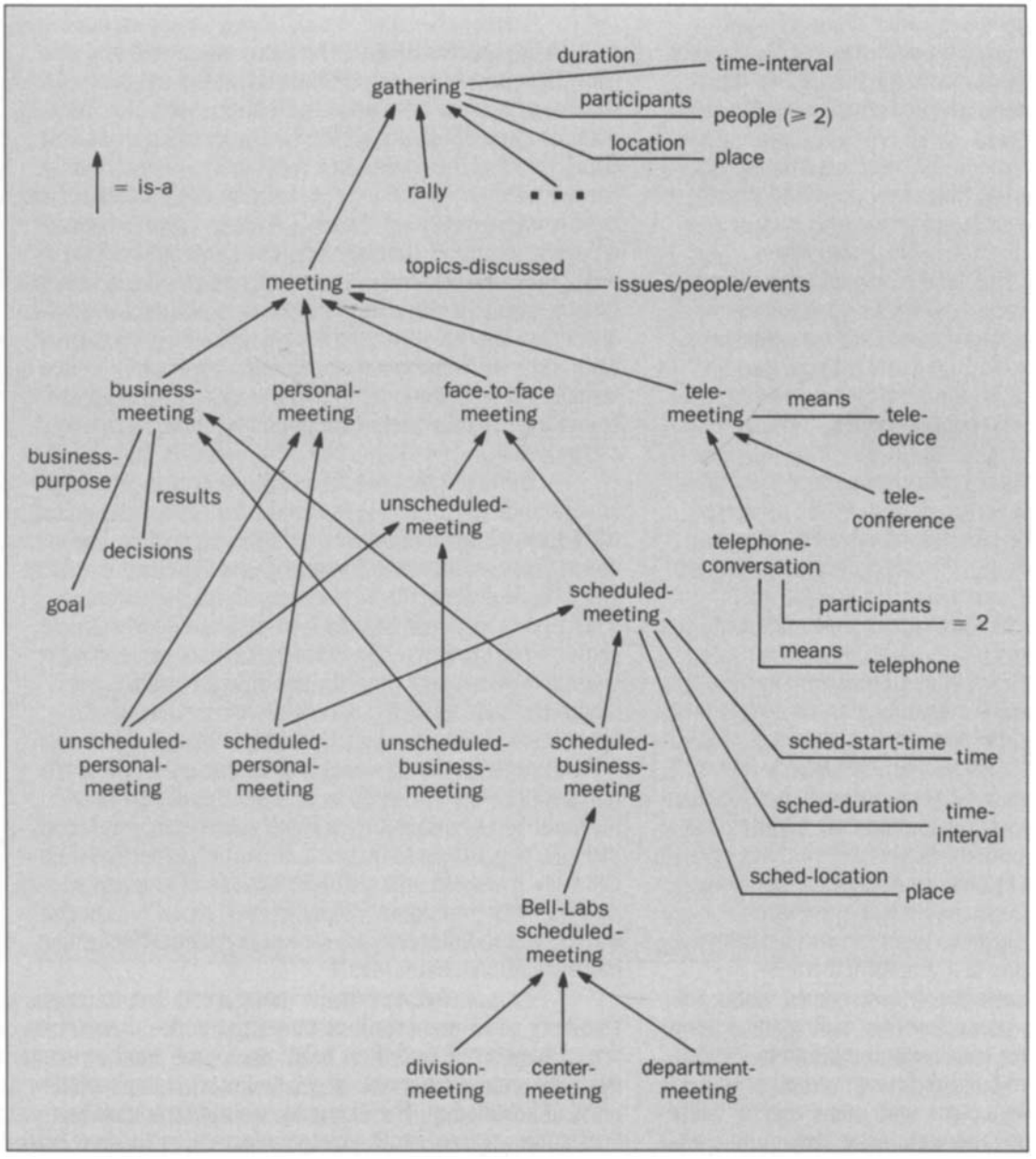


Figure 1. A semantic network.

the sub/superclass relation (often called *is-a*); this link induces a generalization hierarchy over classes. Because of the meaning of generalization, more specific nodes can be considered to "inherit" the properties specifed for their ancestors in the hierarchy.

For example, consider the simple semantic net in Figure 1. This is a partial taxonomy of kinds of gatherings. The *is-a* links connect classes to other classes of which they are specializations. For example, a `business-meeting` is a kind of `meeting`. Since that is the case, `business-meetings` should have all of the properties that `meetings` in general have, e.g., `topics-discussed`, `participants`, `location`, etc. (note that some of these come in turn from `gathering`). These properties are considered to be inherited from nodes higher up in the network. Classes can also further specialize those properties (the `means` of a `telephone conversation` is more tightly restricted than the `means` of a `tele-meeting`), or simply add features that members of the more general class do not universally have (e.g., not all `meetings` have a `business-purpose`, but all `business-meetings` do).

Semantic network systems differ from one another on a host of issues. For example, some allow the "cancellation" of inherited properties. In this way, exceptions to general rules can be crudely represented. Some networks have a single top node; some a single bottom node; some are complete lattices. Some nets allow all link names (relations) to be specified by the user; others predefine a subset (or all) of the binary relations that are the building blocks of the system. And some even have extended notations that allow the representation of all first-order logic expressions in a graphical form.

The typical semantic net system comes with a set of structure-manipulation procedures for adding and deleting nodes and links, and for traversing the graph in various ways. This allows users to build their own search procedures. Semantic nets always come with some core of built-in inheritance mechanisms, although these vary quite a bit in detail and in completeness.

Indeed, because semantic nets so often allow the user to directly manipulate the data structures, and also allow exceptions (thereby undermining the meaning of what appear to be true universal statements, like "all mammals bear their young live"), the completeness and correctness of their inference mechanisms is extremely hard to predict. To the extent that one can determine what logic is implemented in these Tinkertoy-like mechanisms, it does seem to be considerably less general than first-order logic. For example, there is usually no form of negation or disjunction in these notations, and the places where quantification can occur are very limited.[5] There have not been any careful analyses of the complexity of inference in semantic nets in general, although work on "frames" (see below) is related closely enough that results there may carry over.

Semantic net notations are popular in language understanding programs, especially for representing lexical (dictionary) information. Since they have no rule-like component, semantic networks have found little use in expert systems. However, they are especially nice in contexts where recognition of objects is at issue (as, for example, in some vision systems), since they allow convenient representation of complex objects, and the generalization hierarchy lends itself to successive refinement of hypotheses about the identification of objects. Also, the style of organization of code in object-oriented programming systems can be seen to be directly related to semantic network notations. Finally, semantic nets have had a strong influence on work on semantic data models in database management, with the notions of *is-a* and *part-of* figuring very prominently in most such models. (Panel 2 addresses the differences between knowledge bases and more traditional databases.)

Frames. While semantic networks seem to emphasize very small and primitive units (the nodes themselves are unstructured) and their relations to one another, frame systems emphasize much larger and more complicated units of knowledge. For example, we might postulate a "restaurant frame," with enough information to allow us to recognize and act appropriately in almost any restaurant

Panel 2. KBs versus DBs

Is it possible to use a standard database (DB) as a knowledge base? Much of the function of knowledge bases as described here seems like it could be covered by typical database management systems.

There are a number of quite significant ways in which DBs and KBs differ:

- The complexity of the kinds of facts recorded in a typical knowledge base is far greater than a database can accommodate.
  - —As mentioned in the text, it is usually necessary in AI applications to deal with incomplete information. Databases have almost no mechanism for leaving things unsaid. Every field must be filled with an individual value. When quantified expressions are allowed, it is only as integrity constraints (for checking the DB), and not as stored facts available for retrieval or inference.
  - —Facts stored in a database must conform to a rigid form, and exceptions are not allowed. In contrast, knowledge-based systems usually cover a wide range of inhomogenous knowledge.
  - —Knowledge bases (especially semantic networks and frames) emphasize relations among the generic classes, whereas DBs do not allow relations among the relations to be directly expressed.
- As we have emphasized, knowledge bases depend crucially on an interpretation of the knowledge structures. Databases, on the other hand, explicitly avoid concern with what the data mean or any logical consequences of the data. They are usually only concerned with records, which are purely uninterpreted data structures. One important consequence of this difference is that there is no notion of entailment—and thus inference about the domain—defined for DBs. While databases may allow complex queries to be answered, the computations are only about data structures (counting fields, doing exact matches on strings in fields, etc.).
- Knowledge bases represent generic descriptions in a way that allows new classes or instances to be recognized, or classified under them. Databases are simply flat representations of facts, and do not have conditions for recognizing new items as members of classes to which they have not been explicitly stated to belong.
- The part of a DB that might correspond to concepts or classes in a KB—the schema—stays fixed, whereas the class structure in a typical knowledge base changes fairly often. Typical AI applications have as many—or more—classes (relations) than instances, while DBs are addressed to problems where the instances to be stored greatly outnumber the relations, usually by orders of magnitude.

These differences between databases and knowledge bases are not a matter of black and white. It is indeed possible to consider a relational database as a KB of an extremely simple form (see Reference 12 for details). However, given the complexity and needs of AI applications, this is not likely to be a productive view, nor is the point of view needed to see it this way typical in database management. Database management is concerned mainly with the important issues of scaling up, security, sharing, and distribution. It is much less concerned with the forms of knowledge that can be stored and their logical consequences. Work in knowledge representation is centrally concerned with interpretation and inference.

we encountered. This frame would encode all of our expectations about common restaurant features (e.g., that they serve food, take cash, have waiters, have tables, etc.) as well as details of particularly memorable experiences we have had with unorthodox restaurants (Legal Seafoods in Boston is usually cited as an example, for its "pay when you order" policy). The frame would include pointers to subframes, which would handle particular kinds of restaurants, such as fast food, cafeteria, elegant, etc. It would also include default assignments for many particular features of restaurants—the features would be captured in "slots" of the frame, and a default would specify a particular value to use for a slot in the absence of any evidence to the contrary, such as the assumption that the right place to

look for descriptions of food is a menu, which you will be given by a host or hostess.

The original conception of frames[16] focused on their use in recognition, especially in visual contexts. Similarly, frames were considered the right style of representation for prediction-based tasks, such as reading a paragraph or story. By setting up predictions based on the general context of the story and the known events, we might get a system to interpret subsequent events and integrate them. This was found particularly useful in accounting for how readers can infer implicit connections among sentences like "John went to a restaurant. He asked the waitress for a hamburger. Eventually, he paid the tip and left." By setting up expectations from the restaurant frame, we can interpret "the waitress," where no such person had been previously mentioned. We can understand the relations of the tip to the waitress and John, and can assume that John received and ate his hamburger and was reasonably pleased with it. While these conclusions are only assumptions, they are of the same sort that we seem to make in everyday commonsense situations.

These are all nice desiderata for a representation and inference system, but they are far from a technical specification. Subsequent to Minsky's original paper, a number of frame system implementations sprang up, with most concentrating on the structure of slots and arranging frames in an *is-a* hierarchy. While these structured frames often allowed default values, they gradually diverged from Minsky's original intent. Some framelike approaches did attempt to remain true to the recognition/prediction goals of the original conception, and a great deal of work was done on prototypical sequences of events in everyday life (like what happens in a restaurant), called *scripts*. A number of systems, mainly used in processing natural language stories, were built on top of the scripts mechanism, with some limited success in dealing with common scenarios by invoking complex knowledge of previously encountered events.

Panel 3. A Frame for a Department Meeting

```
11257-DEPARTMENT-MEETING
is-a:                MEETING
attendee:            at-least 3 11257-DEPARTMENT-MEMBER
                                (require: RON)
location:            a ROOM
                                (default: 3D-473)
                                (if-added: (reserve location))
date:                a DATE
                                day: {Mon,Tues,Wed,Thurs,Fri}
agenda:              a list of
                                TOPIC
                                   name:      a TOPIC-NAME
                                   speaker:   a PERSON
expected-duration:   a TIME-INTERVAL
                                (prefer: (less-than ONE-HOUR))
actual-duration:     a TIME-INTERVAL
                                (default: (same-as expected-duration))
start-time:          a TIME
                                (prefer: (and (before 4:30-pm)(after 1:30-pm)))
expected-end-time:   a TIME
                                (if-needed: (plus expected-duration start-time))
```

Scripts are sometimes hard to pin down, and their logical and complexity characteristics are unknown, in part because typically the only definition of a script system is its Lisp implementation. On the other hand, while less ambitious, the more structural type of frame has led to some well-defined and broadly used representation systems.

This latter type of frame is defined by a set of superframes and slot restrictions. Superframes specify other descriptions that are more general, and the restrictions specify constraints on the "fillers" of the slots when the frames are instantiated (i.e., when an individual is to be described by one). For example, Panel 3 illustrates what a frame for a department meeting might look like. Such a representation includes

- Absolute restrictions on items that can fill slots (for example, the `actual-duration` of a meeting must be a time interval, and the meeting's `date` must be on a weekday)
- Preferences and defaults for such fillers (e.g., after-lunch meetings are preferred; if no `location` is specified, assume `3D-473`)
- Embedded frames (e.g., `DATE, TOPIC`)
- "Attached procedures" for computation of values (e.g., `expected-end-time`) and consistency maintenance (adding a `location` should cause a change to the `schedule` slot of the frame representing the room).

In many languages (and as in Figure 2) frames mix structures intended to be descriptions (like "a `MEETING` with at least three `DEPARTMENT-MEMBER`s") and assertions (like "a `11257-DEPARTMENT-MEETING` has an `expected-duration` of `ONE-HOUR`"). In the former case, a description specifies necessary and sufficient conditions, which will aid in recognizing or classifying individuals; in the latter case, an assertion gives only necessary conditions, which allow the inference of properties of individuals once appropriately classified. In either case, inheritance of properties from superframes to subframes works much like it does in semantic networks.

Let us for the moment take a view of frames strictly as structured descriptions. In this view a frame is a complex type with internal structure. Pairs of frames can then be considered to have a subsumption relationship—some frames are more general than others, on the basis of their structure. For example, "a `PERSON` all of whose `children` are `DOCTOR`s" subsumes "a `MAN` all of whose `relatives` are `SURGEON`s" (assuming that it can be proven that `DOCTOR` is more general than `SURGEON`, `PERSON` more general than `MAN`, and `relative` more general than `child`), because it is not possible under any interpretation for there to be someone described by the latter description and not the former. This subsumption relationship can be computed just on the basis of the structure of the terms, without recourse to any knowledge of particular individuals. This is akin to finding logical tautologies (like $p \vee \neg p$), which are necessarily true independent of any particular facts about the world.

The fact that subsumption can be computed automatically means that, as new descriptions are entered into a frame system, it can be determined where they should fit with respect to previously specified frames. This *classification* inference, as it is called, has been found to be extremely useful in a number of situations, including keeping a knowledge base consistent when adding new information, providing a more powerful query language for retrieving items that satisfy a description, and allowing more flexibility in invoking rules (by allowing more complex matches to be made on the antecedents). Classification is only available in systems with a strict interpretation of the meaning of frames (i.e., as necessary and sufficient conditions).

Structural frame systems have recently been the subject of mathematical analysis. Elsewhere,[17] we have described a reasonably complex frame language in formal terms, with a formal semantics specified. We have also analyzed the complexity of subsumption determination in a set of simple languages, with some surprising results: in even a very simple frame language, computing subsumption is co-NP-hard (see Panel 4).

Hybrid Reasoning Systems

Each of the representational methodologies dis-

Panel 4: Complexity in a Simple Frame Language

The core of many frame languages can be captured in a simple notation, using a small set of operators for forming descriptions. We might call these ANDC, ALLC, SOME, and RESTRICT, and these examples illustrate their intent:

(ANDC `oil-exec man`)
["an oil executive and a man"]

(ALLC `ranchhand texan`)
["a thing all of whose ranchhands are Texans"]
(SOME `adopted-child`)
["a thing with at least one adopted child"]
(RESTRICT `child female`)
["a female child"]

$\mathcal{TL}^-$

$\mathcal{TL}$

As indicated, let us call the language that includes all four operators $\mathcal{TL}$, and one that includes only the first three $\mathcal{TL}^-$. Although these languages look barely different, it can be shown that there are things that can be said in $\mathcal{TL}$ that cannot be said in $\mathcal{TL}^-$.

Now consider devising subsumption algorithms for these two languages. In the case of $\mathcal{TL}^-$, we could imagine putting expressions in a canonical form that grouped all top-level descriptions under a single ANDC at the top, and grouped together all ALLC and SOME restrictions that applied to the same slot (first argument). Subsumption checking would then just be a matter of walking down the more general structure, making sure that every expression in that frame had a corresponding expression in the more specific one. Since there is no way to chain restrictions together (i.e., you cannot find out something about a slot by looking at restrictions on another slot), we need to walk down the more general structure only once. Indeed, we have devised an algorithm that does this, and have proven that it works in $O(n^2)$ time.

On the other hand, there is no way to compute subsumption for $\mathcal{TL}$ by only walking down one of the structures once. The RESTRICT operator allows a kind of chaining, so we may have to traverse each frame's structure many times. For example,

```
(ANDC person
      (ALLC child doctor)
      (ALLC (RESTRICT child doctor)
      lawyer)
      (ALLC (RESTRICT child lawyer)
      famous))
```

("a person all of whose children are doctors and all of whose doctor-children are lawyers and all of whose lawyer-children are famous") is subsumed by

```
(ANDC person
      (ALLC (RESTRICT child famous)
      doctor))
```

("a person all of whose famous-children are doctors"), but it takes a lot of work to figure it out. We have shown that subsumption for $\mathcal{TL}$ is co-NP-hard, and thus most likely has at best an exponential algorithm (see Reference 5 for complete proofs and formal semantics).

cussed in the previous section has its niche. However, in many cases, a complex problem will call for multiple types of reasoning competence. While it is often possible to force one representation to handle in an awkward way what another handles smoothly, it is usually more desirable to attempt to integrate more than one system into a hybrid system. As the field has gotten a better feel for what production rules, frames, etc., are good—and not so good—for, it has seen a strong trend toward hybrid reasoning systems.

Hybrids can be constructed from any of the standard representation methodologies, as well as more ad hoc schemes geared toward particular narrow applications. In

dividing the representational labor, it becomes much easier to construct a knowledge base, and to try different reasoning strategies on various problems. It also makes it possible to compensate for inadequacies of one approach by appealing to the strength of another. For example, CENTAUR is a hybrid system that integrates production rules and frames in a medical diagnosis context.[18] It uses the frame generalization hierarchy to organize rules into groups relevant to particular diseases. This overcomes the difficulty in dealing with one large undifferentiated rule base (and having to deal with it by creating artificial "context" variables). There are at least two projects at AT&T working on hybrids of rules and frames—MCL and PRESTO. MCL is being used in the Programmer Productivity System (PPS) at Indian Hill Park; the PPS knowledge base now contains approximately 600 MCL rules.

A closer look at the goals of knowledge representation reveals a more fundamental reason for integrating two kinds of representation systems. There are at least two different kinds of knowledge that go into sophisticated reasoning: one that we might call "terminological," having to do with the meanings of the complex terms we use to talk about domains, and the other "assertional," having to do with actual statements of facts using those terms. For example, the relationships between "employee" and "employer" and "job" have to do with knowledge of terms, independent of the existence and identification of any particular objects. On the other hand, "this transistor has a threshold voltage of 0.4 V" is a matter of fact whose truth can be determined only by verification in the physical world. Terminological knowledge tends to be timeless, and complete; it is often concerned with types of objects. Assertional knowledge tends to be relational, and it is often incomplete (e.g., "there is a dog barking out there" does not specify which individual is involved; "the cause of the metabolic acidosis is not shock" does not specify what the cause is).

Of the representation styles of which we have spoken, two fit this split rather nicely. Frames are used for object-centered, structured descriptions. They are rather poor for making general statements, especially incomplete ones. Logic, on the other hand, was created to allow expression of general facts, especially incomplete ones. But in standard logic, all predicates are primitive, and thus unrelated to one another. With this in mind we developed an integrated hybrid system, called KRYPTON,[17] in which a frame component provided complex, interrelated predicates for use in a first-order logic assertional component. KRYPTON was developed at Schlumberger and subsequently improved at Bell Laboratories.[19] Deductions in the assertional part take advantage of relations established between terms in the terminological part. Further, because of special-purpose code for efficient classification in the frame component, some deductions proceed much more rapidly than if the rough equivalents of the frame definitions were encoded directly in first-order logic.[20]

KRYPTON is rather large and cumbersome. In a related project,[21] we created a hybrid system in the spirit of KRYPTON, but with more reasonable computational properties. We used roughly the same frame language, but instead of using first-order logic, we created a very simple assertional mechanism of the sort more typical in other frame systems. The resultant system, called KANDOR, is small and simple and has been found extremely useful in a knowledge-based information retrieval system.[22] In the AI Principles Research Department, we are currently engaged in a redesign and reimplementation of KANDOR, to extend its expressive power somewhat and to make it available internally to interested parties in AT&T. Subsequently we intend to integrate our new system—CLASSIC—with a Horn clause assertional component; we have implemented a frame unification procedure that will facilitate this merger.

Research Topics in Knowledge Representation and Reasoning

Despite the potential leverage to be gained by integrating different types of reasoning components, there are still some fundamental and serious obstacles to achiev-

ing the ideal KR component for knowledge-based systems. In light of this, there are several other current research directions worth noting, although we will not have time to dwell on details.

First, in the kind of problem-solving tasks for which we envision AI systems, it would be virtually impossible to have complete information. When planning a trip, for example, we simply cannot know whether a plane will leave on time, whether our car will start, or exactly what the transit time will be between home and airport. Humans cope with this constant incompleteness by making assumptions, based on their prior experience and on "common sense." The making of assumptions, however, is not a deductive activity, and once an assumption is adopted it can (usually) easily be revoked. This presents a challenge for reasoning systems based in any way on standard logics, since they are *monotonic*: once a conclusion has been reached it will still stand even with the addition of new information. But much of human reasoning is nonmonotonic, in that new information can invalidate old assumptions. As soon as I discover that my battery is dead, I drop the assumption that my car will start—and any conclusions I based on it.

Systems for semantically coherent but nonmonotonic reasoning are one of the hottest research topics in the field. Investigation is proceeding on new, nonstandard logics, especially to deal with defaults and exceptions (both of which lead to nonmonotonicity). More practical work is proceeding on "truth-maintenance" systems that allow assumptions and dependent conclusions to be retracted in a reasonable manner. Etherington[23] and Reiter[24] have written excellent introductions to this research and the problems it is trying to solve.

Second, it appears that in many cases human reasoning looks very little like theorem-proving, and much more like manipulating direct "models" of its domain.[25] Humans seem to do better on problems specifying concrete situations than they do on abstract problems with the same logical structure. All of this leads us to believe that there is an important role to be played by what has been called "model-based" reasoning—using direct database-like, or "vivid"[26] representations of items in the domain. For one thing, manipulating such models would be tremendously more efficient than handling the general cases that logical theorem-provers do. The trick is to find a reasonable way of constructing a vivid model (presumably, default assumptions will almost always be needed), with as little loss of generality as possible (although the tradeoff tells us that some must be sacrificed). We have been investigating this area, with initially slightly discouraging results—even a very simple technique for fleshing out a vivid model with default values is computationally problematic.[27] However, the idea still seems to hold much promise.

Finally, it should be pointed out that while virtually all work in the field assumes that a declarative, propositional knowledge base can be the basis of intelligent behavior, there are those who have expressed reservations about this point of view. While much of the response has been philosophical, there is a small amount of technical work on nonrepresentational problem-solving and planning systems, and a growing amount of work on "connectionist" architectures that claim to do reasoning without the use of symbols. The jury is still out on this work.

Conclusion

Knowledge representation is basically a "backstage" operation. If the representation and reasoning system that supports a knowledge-based system is appropriate, the end user should not even know that it is there. The system will draw the same obvious conclusions that a human would without hesitation. On the other hand, given the prevalence of the knowledge representation hypothesis, it is probably fair to say that KR is a fundamental technology essential for virtually all work in artificial intelligence. Just about all AI researchers and developers do knowledge representation, whether or not they are aware of it or consider it part of their jobs.

We have characterized the KR enterprise as the determination of what semantically coherent information can be extracted from what forms of representation in a

computationally dependable way. The need to have representations be meaningful and interpreted consistently puts the field squarely in the area of logic, albeit the need for computational tractability steers it in a different direction than traditional studies in mathematical logic. Given the problems approached by AI research, we continually face the tradeoff between expressive power and tractability.

As should be evident, knowledge representation takes its inspiration from a wide variety of sources, including philosophy, psychology, and mathematics of computation. Unfortunately there is not yet a textbook that would allow one to explore the entire field in depth. The reader interested in investigating this area further would have to look into a number of papers on individual issues and systems. Perhaps the best resource one can find now is a collection of readings covering the most important research contributions to the field.[28] This collection has a general introduction and each paper is introduced with notes that attempt to show its place with respect to the others.

Finally, we have mentioned some of the types of tools currently popular in AI. These systems are quite useful, and a number of them are being developed and used at AT&T. But there is still a vast amount of research to be done. The human commonsense reasoning that we attempt to emulate is robust and flexible; it is only roughly logical and right enough of the time to get by. Implemented KR systems tend to be rigid and deal only with strictly logical truths. Those that attempt to deal with "fuzziness" fall a long way short of the power of human reasoning, and we have barely scratched the surface of nonmonotonic reasoning. We are just beginning to get below the surface of competent reasoning in a few of its forms.

References

1. B. C. Smith, "Reflection and Semantics in a Procedural Language," Ph. D. Thesis and Technical Report MIT/LCS/TR-272, Massachusetts Institute of Technology, Cambridge, 1982.
2. E. Mendelson, *Introduction to Mathematic Logic*, 3rd ed., Wadsworth and Brooks/Cole Advanced Books and Software, Monterey, Calif., 1987.
3. M. R. Garey and D. S. Johnson, *Computers and Intractibility: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, 1979.
4. H. J. Levesque, "Knowledge Representation and Reasoning," *Annual Reviews of Computer Science*, Vol. 1, Annual Reviews, Inc., Palo Alto, Calif., 1986, pp. 255-287.
5. H. J. Levesque, and R. Brachman, "Expressiveness and Tractability in Knowledge Representation and Reasoning," *Computational Intelligence*, Vol. 3, No. 2, May 1987, pp. 78-93.
6. R. J. Brachman, " 'I Lied about the Trees' or, Defaults and Definitions in Knowledge Representation," *The AI Magazine*, Vol. 6, No. 3, Fall 1985, pp. 80-93.
7. D. Etherington and R. Reiter, "On Inheritance Hierarchies with Exceptions," *Proceedings*, AAAI-83, American Association for Artificial Intelligence, Washington, D.C., August 1983, pp. 104-108.
8. J. McCarthy, "Programs with Common Sense," *Semantic Information Processing*, M. Minsky, ed., The MIT Press, Cambridge, Mass., 1968, pp. 403-418. Reprinted in *Readings in Knowledge Representation*, R. J. Brachman and H. J. Levesque, eds., Morgan Kaufmann Publishers, Los Altos, Calif., 1985, pp. 300-307.
9. H. A. Kautz and E. P. D. Pednault, "Planning and Plan Recognition," *AT&T Technical Journal*, Vol. 67, No. 1, January/February 1988, pp. 25-40.
10. J. Hirschberg, B. W. Ballard, and D. Hindle, "Natural Language Processing," *AT&T Technical Journal*, Vol. 67, No. 1, January/February 1988, pp. 41-57.
11. J. A. Robinson, "A Machine-Oriented Logic Based on the Resolution Principle," *Journal of the Association for Computing Machinery*, Vol. 12, No. 1, 1965, pp. 23-41.
12. H. J. Levesque and R. J. Brachman, "Knowledge Level Interfaces to Information Systems," *On Knowledge Base Management Systems*, M. L. Brodie and J. Mylopoulos, eds., Springer-Verlag, New York, 1986, pp. 13-34.
13. M. R. Genesereth and N. J. Nilsson, *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., Los Altos, Calif., 1987.
14. T. J. Kowalski, "The VLSI Design Automation Assistant: A Synthesis Expert," *AT&T Technical Journal*, Vol. 67, No. 1, January/February 1988, pp. 81-92.
15. G. T. Vesonder, "Rule-Based Programming in the UNIX® System," *AT&T Technical Journal*, Vol. 67, No. 1, January/February 1988, pp. 69-80.
16. M. Minsky, "A Framework for Representing Knowledge," *The Psychology of Computer Vision*, P. H. Winston, ed., McGraw-Hill Book Company, New York, 1975, pp. 211-277.
17. R. J. Brachman, V. P. Gilbert, and H. J. Levesque, "An Essential Hybrid Reasoning System: Knowledge and Symbol Level

Accounts of KRYPTON," *Proceedings*, 1985 International Joint Conference on Artificial Intelligence, Los Angeles, August 1985, pp. 532-539.
18. J. Aikins, "Prototypical Knowledge for Expert Systems," *Artificial Intelligence*, Vol. 20, No. 2, 1983, pp. 163-210.
19. P. Devanbu and R. Brachman, "KRYPTON: The User Interface," AT&T Bell Laboratories Technical Memorandum, January 1987.
20. M. E. Stickel, "Automated Deduction by Theory Resolution," *Proceedings*, 1985 International Joint Conference on Artificial Intelligence, Los Angeles, August 1985, pp. 1181-1186.
21. P. F. Patel-Schneider, "Small can be Beautiful in Knowledge Representation," *Proceedings*, IEEE Workshop on Principles of Knowledge-Based Systems, Denver, December 1984, pp. 11-16. Extended version appears as AI Technical Report No. 37, Schlumberger Palo Alto Research, Palo Alto, Calif., October 1984.
22. P. F. Patel-Schneider, R. J. Brachman, and H. J. Levesque, "ARGON: Knowledge Representation Meets Information Retrieval," *Proceedings*, First Conference on Artificial Intelligence Applications, IEEE, Denver, December 1984, pp. 280-286.
23. D. W. Etherington, "Reasoning with Incomplete Information: Investigations of Nonmonotonic Reasoning," Ph.D. dissertation, University of British Columbia, June 1986. Also appears as *Reasoning from Incomplete Information*, Pitman Research Notes in Artificial Intelligence, Pitman, London, 1987.
24. R. Reiter, "Nonmonotonic Reasoning," *Annual Review of Computer Science*, Vol. 2, Annual Reviews, Inc., Palo Alto, Calif., 1987, pp. 147-186.
25. P. N. Johnson-Laird, *Mental Models*, Harvard University Press, Cambridge, Mass., 1983.
26. H. J. Levesque, "Making Believers out of Computers," *Artificial Intelligence*, Vol. 30, No. 1, 1986, pp. 81-108.
27. B. Selman and H. A. Kautz, "The Complexity of Model-Preference Default Theories," to appear in *Proceedings*, Seventh Canadian Conference on Artificial Intelligence, Canadian Society for Computational Studies of Intelligence, Edmonton, Alberta, June 1988.
28. R. J. Brachman and H. J. Levesque, *Readings in Knowledge Representation*, Morgan Kaufmann Publishers, Inc., Los Altos, Calif., 1985.

(Manuscript received October 6, 1987)