Knowledge Bases

Adam Pease
Teknowledge
1810 Embarcadero Rd
Palo Alto, CA 94030
Voice: 650 424 0500
Fax: 650 493 2645
apease@teknowledge.com

Raymond A. Liuzzi, PhD
AFRL/IFTD
525 Brooks Road
Rome, New York 13441
Voice: 315-330-3577
Fax: 315-330-2563
liuzzir@rl.af.mil

David Gunning
Cycorp
3721 Executive Ctr Dr.
Austin, TX 78731
Voice: 512 342 4000
Fax: 512 342 4040
dgunning@cycorp.com

In this paper we present an overview of knowledge-based systems. We begin by discussing the basic characteristics of knowledge base rules and present an example comparing a toy knowledge base to an equivalent procedural version of the same code. We then discuss the applications for which it makes sense to use knowledge-based technology and the advantages offered by reuse of a general-purpose *upper ontology*. Next we describe important research and applications in both knowledge representation and knowledge-based systems. We then describe how work in philosophy and other research has had an impact on knowledge-based systems. We conclude with thoughts on where the field is heading.

Introduction

A *knowledge base* is a collection of rules and statements that describe a portion of the world. A knowledge base is interpreted by an *inference engine*. The inference engine executes when the human user or another software system poses a query. In trying to answer the query, the inference engine dynamically combines *rules* and *facts*.

As a very simple example, let's say we have in our knowledge base

Rule: `If ?X is a man then ?X is mortal.`

Fact: `Socrates is a man.`

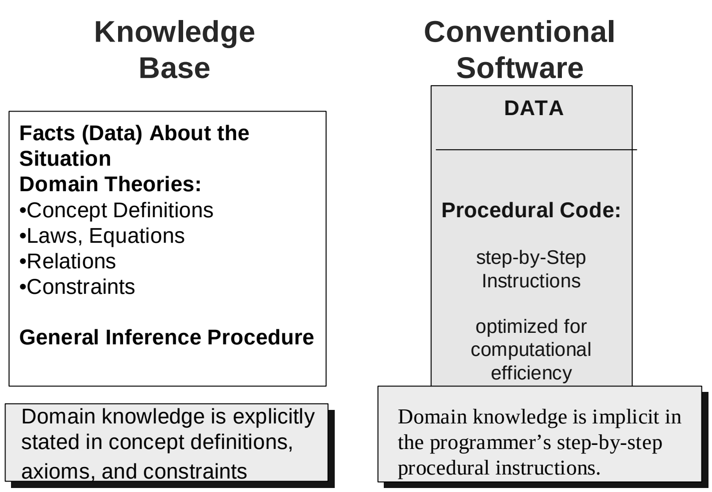

Figure 1 - Knowledge Base vs Conventional Software

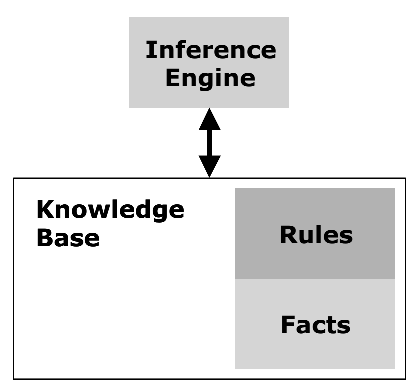

Figure 2 - Knowledge Based System Components

`G->E`
`G->D`
`F->C`
`F->B`
`D+E->A`
`B+C->A`
	`var A,B,C,D,E,F,G`
	`if F then`
	`  B and C = true`
	`if B and C = true then`
	`  A`
	`if G then`
	`  D and E = true`
	`if D and E = true then`
	`  A`

Figure 3 - Example Rule Base	**Figure 4 - Equivalent Procedural Code**

We can ask the query "**`Is Socrates mortal?`**". The inference engine combines the rule and the fact to conclude, "**`Socrates is mortal`**."

Any large knowledge base is likely to consist of several levels. The top level is often called an *upper ontology*. An upper ontology consists of the most general terms and concepts and their definitions, relationships and constraints. These include terms such as tangibility and set membership. At the next level are *core theories* that consist of general knowledge about the world including theories for space, time, causality, motion, belief, and objects described in a formal logical language. At lower levels, the knowledge base becomes more and more *domain specific*. The knowledge reflects the needs of a particular application like airplane part design, diagnosis of skin diseases or Army battlefield planning.

In a knowledge base, the domain knowledge is explicitly stated in formal definitions, axioms and constraints. In a database, the domain knowledge is largely implicit in the programmer's step-by-step procedural instructions.

Conventional *procedural* software also has facts and if/then statements but they are not combined dynamically. Knowledge based software is more robust and change resilient than conventional software in representing declarative information because individual facts can be altered without the need to change the process by which those facts are applied.

Consider an example that contrasts a rule-based program with a procedural program. In Figure 3 we have a knowledge base that has a few rules. **`A->B`** means that if **`A`** is true we can assert that **`B`** is also true. In Figure 4 we have pseudo-code for a procedural program. There are two general types of inference procedure, *backward-chaining* and *forward-chaining*. In a backward-chaining expert system, a user might ask if **`A`** is true, the system would then perform a search backward from any rule which might conclude **`A`** in order to find a *proof* for the conclusion **`A`**. In this example, the search would find two rules, **`D+E->A`** and **`B+C->A.`** It would then try to find support for **`D`**, **`E`**, **`B`** and **`C`**. In this knowledge base, if **`F`** and **`G`** had been asserted as true, then the search for a proof would succeed and the system would respond that indeed **`A`** is true. Logic programming systems such as Prolog (Clocksin & Mellish, 1987) employ a backward chaining strategy.

In a forward-chaining expert system, the inference engine would start with all available facts, find rules that had antecedents that were satisfied by those facts and then assert the consequent of those rules. The cycle would start again with finding rules that matched the available set of facts. OPS-5 (Brownston et al, 1985) is a typical example of a forward chaining system. In both the forward and backward chaining versions, the answers reached would be the same. However, depending upon the rule base, one approach may be significantly faster than the other.

In contrast to the rule-based approach, in the procedural programming language version of this example, the process by which the conclusion is reached is completely up to the programmer. The programmer is free to use what would be essentially a forward or backward chaining approach but is also free to implement a process that fails to find the correct conclusions. This problem becomes especially acute as the set of rules grows larger.

What is a knowledge base for?

Knowledge base programming is a very powerful approach for some but not all applications. An area in which this paradigm is not appropriate would be one in which there is a great deal of numeric computation. Knowledge based programming is appropriate when there is a human expert who can articulate information about his decision making process as rules of thumb. General reasoning, problem solving and system integration are typical uses.

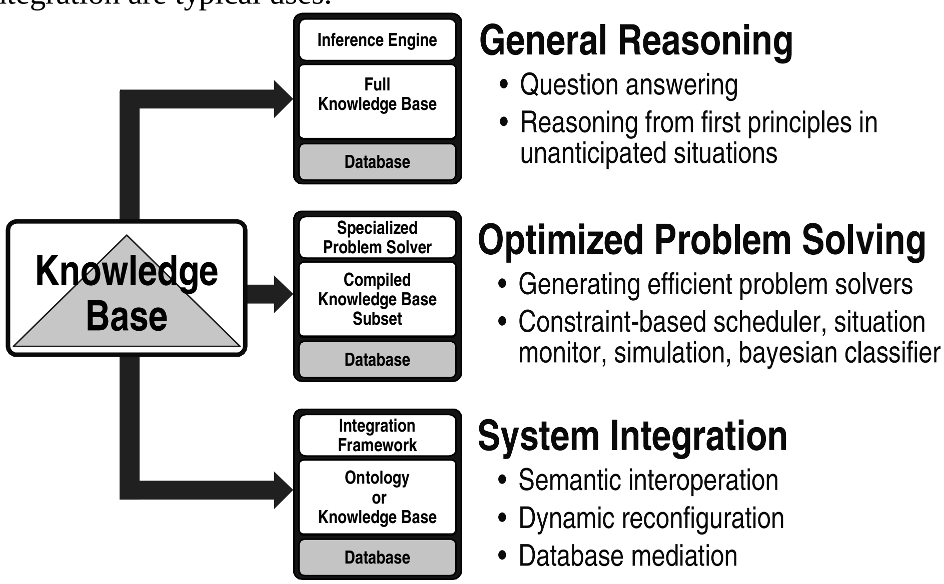

Figure 5 - Applications for Knowledge Bases

Knowledge bases can be used in a reasoning process to infer results directly. For instance a knowledge base can actively inform, diagnose, monitor, etc. If we ask a question like “Which air vehicles can move to a specific region?” a data base answer is found in a very different way from a knowledge base. For a database, a procedure must be written to determine the answer. However, a knowledge base can directly infer the answer from its set of facts and rules.

Upper Ontology

A key issue in modeling information and system integration is whether that modeling is done strictly “bottom up” or with significant “top down” structuring. Typical software projects define a restricted “world” from the ground up. This involves taking inventory of what concepts and relations are relevant and then creating a database or Java classes that cover those concepts and relations. One set of problems with this approach occurs during maintenance. Maintenance is hard when the “world” changes and very new and different assumptions are introduced.

Early expert systems were built from scratch to support particular needs. It was found that when people attempted to reuse those systems for slightly different applications, they ran into great difficulties. The systems were very specific and did not include the more general abstractions needed to use the existing rules in a new way. Knowledge engineers had built in too many specific, simplifying assumptions into the rules. A solution for this problem has been the creation of *ontologies* that are generalized collections of concepts. When expert systems are defined in terms of the most general concepts, they become more robust and able to accommodate changes.

For example, let’s consider hypothetically the construction of a warehouse logistics planning application. Early in the development lifecycle we may start off by assuming that all things in the plan are physical and have a location and we create a database schema based on that assumption. The database may have a table that has position coded as warehouse building number and a bin number and foreign keys in every other table that point to the position table. Later, as requirements evolve, we may want to extend the representation to include tasks so as to manage not only inventory but also shipping. Tasks do not have position and the database will need to be altered. We may now assume that all things have a temporal extent. Later purposes might be added in order to prioritize shipments according to customer needs. Purposes have neither temporal nor spatial extent. The existence of an upper ontology can force the designer to make decisions early on about fundamental issues such as whether entities must have physical or temporal extent. This structuring saves the system from drastic redesign later on. This is somewhat analogous to using an object oriented modeling language to help anticipate design changes and create a robust architecture in software design.

As a further illustration of the utility of an upper ontology, let us consider another hypothetical application. Suppose that we have a system that supports military planning. Our hypothetical system developer has created a database to support the planning system. It contains information about bombs, missiles, tanks, explosives, guns and other equipment. The developer has created a database with an equipment table that contains columns for equipment name and quantity.

Suppose that there is a mission to destroy two enemy bridges and the planning application is employed to help. Let’s say that the supply group for this mission is quite limited and has only one five-pound package of C4 plastic explosive. Each bridge requires two pounds of C4 for demolition. Our hypothetical application will fail because the database says that there’s only one package of C4 and the mission requires two. While it is obvious to a human that the C4, unlike a missile, can be divided, it is not obvious to the software. The designer has built in a faulty assumption that there are simply countable things in the domain. He has neglected to include information about the weight or volume of these things and whether they are divisible.

The distinction between stuff and object has been discussed by philosophers for thousands of years and information designers should take advantage of this knowledge. Stuff is defined as anything that can be subdivided (up to some point) and still retain its characteristics. Examples are water or wood. Objects, or "countable things", cannot be subdivided and retain their identity. Examples include table or chair. There are many boundary cases that have also been formalized.

If our hypothetical designer had been basing his system on an upper ontology, he would have been forced to make a decision about whether the things in the database were Stuff or Object, and he would have been informed about the characteristics that distinguish these two types. As in any simple example, we certainly could conceive of a system which makes this distinction correctly, but the problem is that there are hundreds if not thousands of such distinctions which can make a software system brittle if not addressed.

The Cyc project (Lenat, 1995) and the IEEE SUO (IEEE, 2001) are efforts to create just such an upper ontology.

Research overview

Modern knowledge based research can be broken down into a number of subfields. Some, like AI planning are related but focus more on algorithms for planning than general-purpose knowledge representation. Areas that are relevant for this paper include knowledge base content (ontology), information integration, inference (theorem proving) and knowledge representation languages.

The largest effort by far to create a general-purpose knowledge base is Cyc. The DARPA High Performance Knowledge Bases effort (Cohen et al, 1998) (Pease et al, 2000) used Cyc as well as other much smaller knowledge bases to solve problems in reasoning about international crisis situations and Army battlefield reasoning. The IEEE SUO effort is an effort to create a public open standard for the most general level of an ontology.

There are a number of systems for knowledge based browsing and editing. These include Cyc, GKB (Paley et al, 1997), Ontolingua (Farquhar et al, 1996), and Ontosaurus (Swartout et al, 1996). All these systems support creation, viewing and saving of large knowledge bases in a particular formal representation language. All offer some degree of checking for consistency of the knowledge.

Another area of research in knowledge-based systems is the use of knowledge bases for semantic interchange and integration. DARPA's DAML (DAML, 2001) program is attempting to use ontologies to label information on the World Wide Web with meaning instead of simply formatting or presentation information. KIF (Genesereth, 1991) was an effort to create a very expressive language for representing and sharing knowledge.

Some of the most practical efforts to improve the speed of inference on large knowledge bases have been in what are known as "deductive databases". Relevant efforts include the Warren Abstract Machine (Ait-Kaci, 1991) and its implementation in XSB Prolog (Rao et al, 1997). Efforts to provide reasoning capability directly for expressive knowledge representations include Cyc (Lenat, 1995), LOOM (MacGregor, 1994), SNARK (Stickel et al, 1994), Otter (McCune & Padmanabhan, 1996) and

ATP/JTP (JTP, 2001). Commercial efforts to create expert system shells include ART/KEE, Exsys (Exsys, 2001), M4 (Teknowledge, 2001), and G2 (Gensym, 2001).

AI and Object Oriented Programming

Object oriented programming was the result of merging some simple elements of knowledge bases with conventional procedural programming. Early AI research is responsible for the genesis and popularization of object-oriented programming (OOP) languages (Russel 1995). Craig Hewitt created a new programming concept for *actors* in his PLANNER (Hewitt, 1969) system. The new concepts for *actors* (Greif 1975), along with ideas from a new simulation language, Simula (Dahl and Nygaard, 1966), led Alan Kay to create the first complete object-oriented language, Smalltalk-76 (Ingalls 1978). At about this same time, Marvin Minksy published his classic paper, proposing *frames* as basic knowledge-representation technique in AI (Minksy 1975). These new ideas led to an explosion of object-oriented, and frame systems in AI. Throughout the late 70's and 80's, AI researchers and practitioners explored and popularized the new object-oriented programming concepts of class inheritance and message passing, which eventually evolved into mainstream programming languages such as C++ and later, Java.

Applications

Two of the first expert systems developed were DENDRAL (Buchanan et al, 1969) and MYCIN (Buchanan & Shortliffe, 1984). DENDRAL (which originally stood for DENDRitic Algorithm) was designed to help chemists determine the structure of molecules, a problem previously done painstakingly by trial and error and relying on the expertise of the chemist. DENDRAL worked very well until the number of rules and logic grew beyond a certain point of complexity, when it became very difficult to add new rules or make adjustments to existing ones while maintaining stability. The system essentially became chaotic, with a small change in initial conditions having large and unforeseen impacts down the line.

MYCIN (the name is derived from a common suffix of antibiotic names), designed to diagnose infectious blood diseases, went some way toward overcoming this shortcoming by separating the rules from the inference engine. Also notable was Amarel's (Amarel, 1968) software that attempted to diagnose diseases based on explicit knowledge of the causal mechanisms of the disease process.

DENDRAL's rules and conclusions were categorical. Medicine however, is a profession that, for all its high-tech gadgetry, still relies heavily on physician intuition or heuristic decisions. MYCIN, appropriately for a medical expert, handled uncertainty in its decisions. Its answers would not be straight "Yes" or "No", but "There's a 63% certainty that the patient has X infection Y."

A fundamental advance represented by MYCIN over DENDRAL was that its knowledge base was separated from the control structure. All modern expert systems use this two-part structure, which facilitated the development of expert system "shells" -- a control structure plus empty slots into which one could feed expert knowledge from any domain. The primary difference among the large number of modern expert systems is not how they reason -- they all reason in pretty much the same way. The difference, rather, is in what they know. One expert system may know about infectious diseases, another about oil-bearing rock formations.

The hardest part of creating a new expert system is transferring knowledge from a human expert into the system's knowledge base. In education, we call this "teaching." In AI, it's known as "knowledge engineering." Software that supports the creation of knowledge is known as “knowledge acquisition” software. One of the first knowledge acquisition tools was TEIRESIAS (Davis, 1980), a program that helped the human expert spot gaps and inconsistencies in the knowledge being transferred to the system.

DENDRAL and MYCIN were significant advances for AI in an academic and scientific sense, and while DENDRAL saw some actual applied use, there were a numbers of factors that prevented greater adoption by users in their respective domains. Those factors included “political” issues of user acceptance, degree of domain coverage, speed and maintainability.

One of the first real-world applications of expert system technology was Digital Equipment Corporation (DEC)'s XCON (McDermott, 1988) – an "Expert Configurer." XCON helped DEC salespeople decide what configuration of hardware components was best for a given customer's needs (DEC sold "clusters" of minicomputers that could be configured in hundreds of different ways). XCON then helped DEC production engineers put the components together. XCON was credited with making DEC profitable. But like DENDRAL and MYCIN before it, XCON too would eventually become bogged down as it grew in size and complexity. Many other expert systems are discussed in (Hayes-Roth et al, 1983).

After this early period, companies like Inference, G2, Teknowledge, Exsys and others (cited above) created expert system shells (in some cases built on code in the earlier applications) that provided a general platform for expert systems development.

Many modern systems utilize combinations of techniques, where the expert system is just one component of a system that may include procedural programming, a relational database and even other AI techniques such as neural networks (Kartalopoulos, 1996).

Philosophy and Knowledge Based Systems

Some philosophers anticipated logical methods in AI. Aristotle laid many of the foundations and the mediaeval 'scholastics' like Duns Scotus, William of Ockham and Peter of Spain (later a pope) made important contributions. Gottfried W. Leibniz, inspired partly by the conceptual combinations in the wheels and diagrams devised by the 13th-Century theologian Ramon Llull, and partly by the reputed combinations of ideas in Chinese characters, described a way to use prime numbers to represent basic concepts, and to combine the concepts by multiplying the numbers. Similar ideas occurred to people designing written and spoken 'universal philosophical languages', like George Dalgarno and Bishop John Wilkins in the 17th Century. Leibniz called early for AI when he said that disputing scholars could sit down and, instead of arguing, say "Let us calculate!"

The technical mastery of the needed logic (for properly combining concepts) had to wait for the development in the 1870's and 1880's of predicate logic, especially quantified predicate logic with the quantifiers "all" and "some", developed independently by logicians Gottlob Frege in Germany (Frege, 1879) and Charles S. Peirce in America (Peirce, 1870). This included the logic and algebra of relations, first developed by Augustus De Morgan and Peirce, and later by logicians Ernst Schroeder, Giuseppe Peano and Alfred Tarski. That also formed the basis of the influential logical system in Principia Mathematica by Alfred N. Whitehead and Bertrand Russell (Whitehead & Russell, 1910).

With the development of electronic computers, the possibility of AI occurred to people from the start. In the 1950's, computer pioneer Alan Turing predicted real AI by year 2000. After John McCarthy, Alan Newell, Marvin Minsky and others founded the field of AI in the 1960's, it took a while to re-discover some of the relevant earlier work of the philosophers such as Peirce. Meanwhile various extensions of logic, especially modal and temporal logic, were seen to be relevant to AI. AI itself was the inspiration for so-called "non-monotononic logics" (McDermott & Doyle, 1980), which attempt to formalize defeasible inference (i.e. with exceptions or overriding). When the field of 'ontological engineering' became prominent in the 1990's (dealing with the content, rather than the logic, of knowledge bases), suddenly a vast amount of philosophical theory became relevant to the modelling of concepts in the real world; this included the ontologies of Bolzano, Husserl, Peirce, Whitehead, Lesniewski, Quine and others, and later work in specific ontological domains such as mereology (parts and wholes), models of emotions, individual identity, and formal semiotics (signs and representation).

An excellent summary of the history of philosophy leading to AI is found in (Sowa, 2000).

Related work

Modern knowledge base development draws upon a number of disciplines including linguistics and philosophy. Linguists often divide their study into syntax (language structure), semantics (meaning of phrases and sentences), and pragmatics (world knowledge). It is the study of semantics and pragmatics that has the most to offer AI. One important system is Wordnet (Feldbaum, 1998) that could be characterized as a lexical ontology. It attempts to provide distinct meanings for each word in the English language and a simple set of relations between words such as synonym-antonym, and general-specific. Semanticists have conducted especially important research on representing events and actions in logic (Davidson & Harman 1972) (Parsons, 1990).

Early work in AI used logical deduction to solve many different problems. Early work in AI planning (Green, 1969) treated planning as theorem proving. Subsequent systems used special purpose systems called planners to accomplish the same goal in a more efficient way. Many of the same issues in knowledge representation that are relevant to knowledge base systems are relevant to the AI planning community, such as reasoning about causality and temporal reasoning.

MYCIN, and other expert systems that use *certainty factors* have algorithms for combining and calculating degrees of belief in facts. Those algorithms are *heuristic* and do not have a strong theoretical basis. A related field of research that does have a solid mathematic basis is Bayesian Networks (Pearl, 1988). Another method for dealing with uncertainty is the Dempster-Shafer (Yaeger et al, 1994) approach. A related, but different issue is how to deal with vagueness or imprecision in information. Fuzzy Logic (Zadeh, 1965) is one set of methods for dealing with this issue.

Case-based reasoning (Hammond, 1989) (Reisbeck & Schank, 1989) (Kolodner, 1993) (Aamodt, 1994) (Althoff, 2001,1999a) (Bartsch-Spörl, 1999) (Günter, A. & Kühn, 1999) is another area where declarative knowledge representations are used. However, instead of performing logical deduction on that knowledge, CBR systems perform *matching* and *adaptation*. Also, instead of storing knowledge as rule, CBR systems store knowledge as *cases* that represent particular or canonical sets of events or facts. Each problem that the CBR system is asked to solve is presented as a case or situation that must be matched with prior cases. Information in the old case that is not present in the new case may be adapted and presented as a hypothesis for information that is or may become true in the new situation. An area related to CBR, but more heavily motivated by cognitive science research is analogical reasoning (Vosniadou & Ortony, 1989).

Other related areas include model-based reasoning, qualitative reasoning (Forbus, 1997), planning (Allen, et al 1990), software engineering (Studer, 1999) (Motta, 2000) and logic programming (Lloyd, 1987) (van Hentenryck, 1990). Knowledge based reasoning has also been heavily influenced by research in cognitive science (Bobrow & Collins, 1975) (Newell, 1990). A field with related goals but different technologies is that of knowledge management (Liebowitz, 1999).

Conclusions and Future work

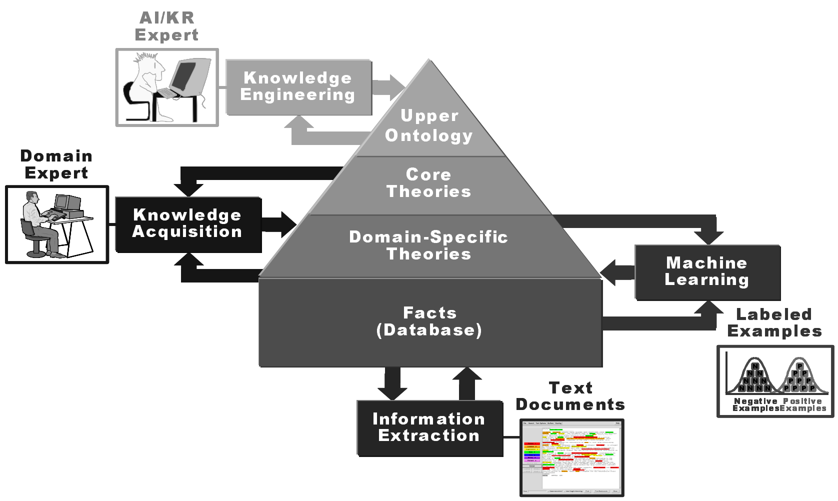

Figure 6 - Knowledge Base Creation Methods

Much has been written about the “Knowledge Acquisition Bottleneck”. It is hard to create formal knowledge suitable for computer processing from the experience and training of human beings. A critical area for future research is in enabling a broader range of people to create formally specified knowledge. There are significant advances being made in four techniques for creating knowledge:

(1) Knowledge Engineering by an AI or logic expert. While this has always been the primary source of knowledge creation, new entry tools and KB organization schemes have made it possible of an AI expert to be significantly more productive by reusing large libraries of KB modules, instead of creating every piece of knowledge by individually.

(2) Knowledge Acquisition from a Domain Expert. New technology is now being developed which will enable AI-novices to enter their knowledge by using natural language dialogs, giving illustrative examples, or by describing analogies.
(3) Knowledge Extraction from Text Documents. As language understanding technology improves, our ability to extract simple facts and pieces of knowledge from text documents is improving to the point that we will soon be able to populate large knowledge-bases with simple facts automatically.
(4) Machine Learning from Observations. Machine Learning technology will give us the ability to fill in the KB with rules and other knowledge, by learning that knowledge from data examples.

This is in fact a focus of the DARPA Rapid Knowledge Formation program (RKF, 2001). The central objective of the Rapid Knowledge Formation (RKF) program is to enable distributed teams of subject matter experts (SMEs) to enter and modify knowledge directly and easily, without the need for specialized training in knowledge representation, acquisition, or manipulation. The resulting knowledge bases (KBs) will be available to provide specific answers to questions and could be applied in many different problem-solving situations. Individual SMEs, as well as different SME communities, must be able to create, exchange, modify, and maintain their own specialized knowledge even where portions of such knowledge supercedes or contradicts other SMEs.

The most difficult technical challenges involve bridging the chasm between human logic and the formal logic of machines. Humans possess vast amounts of common sense acquired through years of experience, employ analogies, situational contexts, and reason using natural language and complex visualizations. Machines are constrained to techniques of mathematical logic, require precise representations and usually employ deductive theorem proving. In addition, the inherent complexity of managing and utilizing large knowledge bases grows exponentially. These challenges underlie our strategy as to which research and development technology areas offer the greatest potential for solutions.

There has been a split in knowledge representation research from reasoning and inference research. Reasoning research has often addressed issues of theory such as the completeness or soundness of various proof procedures. Knowledge representation work has typically addressed very small knowledge bases. Occasionally, large but very simple knowledge bases have been generated from databases for testing. A crucial area for future research is to combine large knowledge bases that have been created by humans to solve practical problems, and research in inference technology to develop fast techniques for inference on real knowledge bases.

References

Aamodt, A. and Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches. *AI Communications Vol. 7, No. 1*, 39-59.

Ait-Kaci, H. (1991). *Warren's Abstract Machine: A Tutorial Reconstruction.* MIT Press, Cambridge, Massachusetts.

Allen, J.; Hendler, J.; and Tate, A., (eds). (1990). *Readings in Planning*. Palo Alto, CA: Morgan Kaufmann.

Althoff, K.-D. (2001). Case-Based Reasoning. In S.K. Chang (ed.), *The Handbook of Software Engineering and Knowledge Engineering*. World Scientific (to appear)

Althoff, K.-D., Bergmann, R. & Branting, L. K. (eds.) (1999a). *Case-Based Reasoning Research and Development*. Proc. of the Third International Conference on Case-Based Reasoning (ICCBR'99), Seeon Monastery, Germany, July 27-30, Springer Verlag, LNAI 1650

Amarel S. (1968). On representations of problems of reasoning about actions. In Michie D., editor, *Machine Intelligence 3*, volume 3, pages 131-171, Elsevier/North-Holland, Amsterdam, London, New York.

Bartsch-Spörl, B., Lenz, M. & Hübner, A. (1999). Case-Based Reasoning - Survey and Future Directions. In: F. Puppe (ed.), XPS-99: Knowledge-Based Systems - Survey and Future Directions. Springer Verlag, LNAI 1570, 67-89

Bobrow, D., and Collins, A., (eds), *Representation and Understanding: Studies in Cognitive Science*, New York: Academic Press, 1975.

Boole, G., (1854). *Investigations into the Laws of Thought*.

Borgida, A., Brachman, R., McGuinness, D., and Resnick, L., (1989). ``CLASSIC: A Structural Data Model for Objects," in *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data* , pages 59--67, June.

Brachman , R., and Schmolze, J., (1985). "An Overview of the KL-ONE Knowledge Representation System," Cognitive Science, 9(2), April--June, pp. 171--216.

Brownston, L., Farrell, R., and Kant, E., (1985). *Programming Expert Systems in OPS-5: An Introduction to Rule-Based Programming*, Addison-Wesley

Buchanan, B. and Shortliffe, E., editors (1984). *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading, Massachusetts.

Buchanan, B., Sutherland, G., and Feigenbaum, A., (1969), Heuristic DENDRAL: a program for generating explanatory hypothesis in organic chemistry. In Meltzer, B., Michie, D., and Swann, M., editors, *Machine Intelligence 4*. Pages 209-254, Edinburgh University Press, Edinburgh, Scotland.

Clocksin, W., and Mellish. C., (1987). *Programming in Prolog* (3rd ed). Springer-Verlag.

Cohen, P., Schrag, R., Jones, E., Pease, A., Lin, A., Starr, B., Gunning, D., Burke, M., (1998). The DARPA High Performance Knowledge Bases Project, AI Magazine vol. 19 no. 4, Winter.

Dahl, O., and Nygaard, K., (1966). SIMULA--An ALGOL-based simulation language. Communications of the ACM, 9(9):671--678, September.

DAML (2001). DARPA Agent Markup Language, http://wwww.daml.org/

Davidson, D., and Harman, G., (eds) (1972). *Semantics of Natural Language*, Reidel, Dordrecht, Holland.

Davis (1980) - Automated knowledge acquisition tools, Meta-rules: reasoning about control, Artificial Intelligence, 15(3):179-222.

Duda, R., Gaschnig, J., and Hart P., (1979). Model design in the Prospector consultant system for mineral exploration. In Michie, D., editor, *Expert Systems in the Microelectronic Age*, pages 153-167. Edinburgh University Press, Edinburgh, Scotland.

Exsys (2001). Exsys CORVID, http://www.exsys.com/products.htm

Farquhar, A., Fikes, R., and Rice, J., (1996). The Ontolingua Server: a Tool for Collaborative Ontology Construction, In *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge Based System Workshop (KAW95),* Banff, Canada, http://ontolingua.stanford.edu/

Fellbaum, C., (ed.) (1998). *WordNet: An Electronic Lexical Database*. MIT Press.

Fikes, R., Farquhar, A., (1988). Large-Scale Repositories of Highly Expressive Reusable Knowledge; IEEE Intelligent Systems, Vol. 14, No. 2, March/April. Also, KSL Technical Report KSL-97-02

Forbus, K., (1997). Qualitative Reasoning, in *The Computer Science and Engineering Handbook*, A. Tucker (ed), pp 715-733, CRC Press.

Frege, G., (1879) "Begriffssschrift, a Formal Language Modelled upon that of Arithmetic, for Pure Thought," , in van Heijenoort, J. (ed.), *From Frege to Gödel: A Source Book In Mathematical Logic, 1879-1931*. Cambridge, MA: Harvard Univeristy Press, 1967, pp 1-82.

Garnder, M., (1982). *Logic Machines, and Diagrams* (second edition). Chicago, IL: University of Chicago Press.

Genesereth, M., (1991). ''Knowledge Interchange Format", In *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning*, Allen, J., Fikes, R., Sandewall, E. (eds), Morgan Kaufman Publishers, pp 238-249.

Genesereth, M., and Nilsson, N., (1987). *Logical Foundations of Artificial Intelligence*. Palo Alto, CA: Morgan Kaufmann Publishers.

Gensym (2001). *G2 Classic*, http://www.gensym.com/products/g2.htm

Green, C., (1969). Application of Theorem Proving to problem-solving. In *Proceedings of the First International Joint Conference on Artificial Intelligence (IJCAI*-69), pages 219-239, Washington D.C. IJCAII.

Greif, I., and Hewitt, C. (1975). *Actor Semantics of PLANNER-73.* ACM SIGPLAN-SIGACT Conference, Palo Alto, CA, Januray.

Günter, A. & Kühn, C. (1999). Knowledge-Based Configuration - Survey and Future Directions. In: F. Puppe (ed.), XPS-99: Knowledge-Based Systems - Survey and Future Directions. Springer Verlag, LNAI 1570, 47-66

Hammond, K., (1989). *Case Based Planning. Viewing Planning as a Memory Task*. Academic Press.

Hayes-Roth, F., Waterman, D., and Lenat, D., (1983). *Building Expert Systems*. Reading, Massachusetts: Addison Wesley

Hewitt, C., (1969). PLANNER: A language for proving theorems in robots. In *Proceedings of the First International Joint Conference on Artificial Intelligence (IJCAI-69)*, pages 295--301.

IEEE (2001). IEEE Standard Upper Ontology http://suo.ieee.org

Ingalls, D., (1978). The Smalltalk-76 Programming System Design and Implementation, *Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages, January 23-25.*

Kartalopoulos, S., (1996). *Understanding Neural Networks and Fuzzy Logic: Concepts and Applications*, IEEE Press.

Kolodner, J. L. (1993). *Case-Based Reasoning*. Morgan Kaufmann

Lenat, D. (1995). "Cyc: A Large-Scale Investment in Knowledge Infrastructure." Communications of the ACM 38, no. 11 (November).

Liebowitz, J., (1999). *Knowledge Management Handbook*, CRC Press.

Lin, A., and Starr, B., (1998). "HIKE (HPKB Integrated Knowledge Environment) - An Integrated Knowledge Environment for HPKB (High Performance Knowledge Bases)". In *Proceedings of the IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'98)*

Lloyd, J., (1987) *Foundations of Logic Programming* (2nd ed). Springer-Verlag.

MacGregor, R., (1991). The Evolving Technology of Classification-Based Knowledge Representation Systems. In *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, ed J. Sowa, 385-400. San Francisco, Calif., Morgan Kaufmann

McCarthy, J. (1968), Programs with Common Sense. In Minsky, M. L., editor, *Semantic Information Processing*, pages 403-418. MIT Press, Cambridge, Massachusetts.

McDermott, J., (1988). R1: A rule-based configuration of computer systems. Artificial Intelligence, 19 (1): 38-88

McDermott, D., and Doyle, J.,. (1980). Non-Monotonic Logic I. Artificial Intelligence 13, pp 41--72

Minksy, M., (1975). A Framework for Representing Knowledge. In Winston, P., editor, *The Psychology of Computer Vision.* New York, NY: McGraw-Hill.

Motta, E. (2000). Knowledge Level Modelling. In S.K. Chang (ed.), The Handbook of Software Engineering and Knowledge Engineering. World Scientific (to appear)

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., and Swartout, W., (1991). Enabling Technology For Knowledge Sharing, AI Magazine, Volume 12, No. 3, Fall

Newell, A., (1990). *Unified Theories of Cognition*, Harvard University Press.

O'Connor, D., & Barker, V., (1989). Expert systems for configuration at digital: Xcon and beyond. Communication of the ACM, 32 (3), March.

Ontosaurus (2001) Ontosaurus Loom Web Browser http://www.isi.edu/isd/ontosaurus.html

Paley, S.M., Lowrance, J.D., and Karp, P.D. (1997) "A Generic Knowledge-Base Browser and Editor," In *Proceedings of the 1997 National Conference on Artificial Intelligence,* Providence, RI.

Parsons, T., (1990). *Events in the Semantics of English: A Study in Subatomic Semantics*, MIT Press.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.

Pease, A., Chaudhri, V., Lehmann, F., and Farquhar, A., (2000), Practical Knowledge Representation and the DARPA High Performance Knowledge Bases Project, In *Principals of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR-2000).*

Peirce, C. (1870). Description of a notation for logic of relatives, resulting from an amplification of the conceptions of Boole's calculus of logic. *Memoirs of the American academy of arts and sciences*, 9:317-378

Riesbeck, C., and Schank, R., (1989), *Inside Case–Based Reasoning*, Laurence Erlbaum Assoc, Pub.

RKF, (2001). Rapid Knowledge Formation project, http://projects.teknowledge.com/RKF

Russel, S., and Norvig, P., (1995). *Artificial Intelligence, A Modern Approach*. Upper Saddle River, NJ: Prentice-Hall, Inc.

Sowa, J., (2000). Knowledge Representation, Logical, Philosophical, and Computational Foundations. Pacific Grove, CA: Brooks/Cole.

Studer, R., Fensel, D., Decker, S. & Benjamins, V. R. (1999). Knowledge Engineering - Survey and Future Directions. In: F. Puppe (ed.), XPS-99: Knowledge-Based Systems - Survey and Future Directions. Springer Verlag, LNAI 1570, 1-23

Swartout, B., Patil, R., Knight, K. and Russ, T. (1996). Ontosaurus: a tool for browsing and editing ontologies . Gaines, B.R. and Musen, M.A., Ed. *Proceedings of Tenth Knowledge Acquisition Workshop*. pp.69-1-69-12 (http://ksi.cpsc.ucalgary.ca/KAW/KAW96/swartout/ontosaurus_demo.html)

Teknowledge, (2001), M.4. http://www.teknowledge.com/M4/

van Hentenryck, P., (1990). *Constraint Satisfaction in Logic Programming*. MIT Press.

Vosniadou, S., and A. Ortony (eds) (1989). *Similarity and Analogical Reasoning*, Cambridge: Cambridge University Press.

Whitehead, A.N., and Russell, B., (1910) *Principia Mathematica*, 2nd Ed., Cambridge, UK: Cambridge University Press, 1925.

XSB, (2001). http://xsb.sourceforge.net/

Yager, R., Kacprzyk, J., and Pedrizzi, M., (Eds.) (1994), *Advances in the Dempster-Shafer Theory of Evidence*, Wiley, N.Y.

Zadeh, L., (1965). *Fuzzy logic and its applications*, Academic Press, New York.