

# Meaning and Links

*William. A. Woods*

■ This article presents some fundamental ideas about representing knowledge and dealing with meaning in computer representations. I will describe the issues as I currently understand them and describe how they came about, how they fit together, what problems they solve, and some of the things that the resulting framework can do. The ideas apply not just to graph-structured “node-and-link” representations, sometimes called semantic networks, but also to representations referred to variously as frames with slots, entities with relationships, objects with attributes, tables with columns, and records with fields and to the classes and variables of object-oriented data structures. I will start by describing some background experiences and thoughts that preceded the writing of my 1975 paper, “What’s in a Link,” which introduced many of these issues. After that, I will present some of the key ideas from that paper with a discussion of how some of those ideas have matured since then. Finally, I will describe some practical applications of these ideas in the context of knowledge access and information retrieval and will conclude with some thoughts about where I think we can go from here.

Semantic networks, in which nodes are connected to other nodes by relationships called links, are widely used to represent knowledge and to support various algorithms that operate on that knowledge. In 1975, I published a paper titled “What’s in a Link” (Woods 1975) in which I argued for rigor in understanding the intended semantics of such links. Since then, I have attempted to work out a framework and a system for dealing rigorously with semantics in link-based representations and for efficiently organizing and retrieving information using such representations. This is an evolving activity in which many subtle (and some not-so-subtle) issues need to be addressed and adequately provided for. It’s like solving a large puzzle in which many of the necessary

pieces do not yet exist and so need to be invented. Moreover, the remaining pieces are mixed together with many distracting and misleading pieces that need to be recognized as such and set aside as pieces of other puzzles. In this article I will present some of the pieces of this puzzle, as I currently understand them, and describe how they came about, how they fit together, what problems they solve, and some of the things that the resulting framework can do. These ideas apply not just to explicitly link-based representations but also to many similar representations, referred to variously as frames with slots, entities with relationships, objects with attributes, tables with columns, and records with fields, and to the classes and variables of object-oriented data structures. I invite the reader to join me in the continuing evolution of these ideas.

This article expands on an invited talk I presented at the 2004 Principles of Knowledge Representation and Reasoning conference in Whistler, Canada. It was one of two inaugural talks in a new series on “Great Moments in Knowledge Representation.” I was asked to talk about the paper “What’s in a Link”—how it came to be, what was happening in the field at the time, and how the ideas have evolved since then. This gave me an opportunity to think about a number of ideas that led up to the writing of that paper—how they relate to each other and how they continue to develop. This article provides an opportunity to continue those thoughts.

Two of my fundamental assumptions will quickly become apparent: first that notations matter (and I hope to give you a feeling for why this is so), and second, that subsumption and generality play an important role in human-level reasoning. With respect to the first, I ask you to speculate how much of modern mathematics would have happened if we had been constrained to try it in roman numerals. In another dimension, you may be aware

of algorithms that have different complexity if numbers are represented in unary notation rather than binary or decimal. In this article, I will illustrate how some key reasoning algorithms, and in particular how subsumption algorithms, can depend on the notations that we use to represent knowledge and how a proper combination of representation and algorithm can answer some of the following questions: How does a reasoning system find relevant pieces of information and relevant rules of inference when it knows millions of things? How does it acquire and organize millions of items of information? How does it integrate new information with previously existing information? How does it use its knowledge to impose structure on situations?

I will start by describing some background experiences and thoughts that preceded my writing the “What’s in a Link” paper. These provided the perspective that I brought to the task. After that, I will present some of the key ideas from the paper and a discussion of how some of those ideas have matured since then. Finally, I will describe some practical applications of these ideas in the context of knowledge access and information retrieval and will conclude with some thoughts about where I think we can go from here.

## A Perspective on Human-Level Reasoning

It appears that there are two kinds of reasoning that people do. I will characterize them roughly as rule-based and associative. When we’re doing the former, we’re aware of it, and it has steps that we can describe. We know we’re doing something, and if it’s complicated enough, we know that we’re doing work and that we might make mistakes. When we do associative reasoning, however, it’s largely below the level of consciousness, and it appears effortless. It takes clever psychological experiments to reveal how much work actually goes on at this subconscious reasoning level.

It’s probably not an accident, therefore, that the structure of computer programs and the characteristics of logical formalisms tend to resemble the rule-based reasoning that we’re aware of, and we’ve been relatively successful (modulo the combinatorics) at getting computers to do this kind of reasoning. On the associative side, however, it is much harder even to understand what people do, much less figure out how to get computers to do something equivalent to it.

When it comes to dealing with large amounts of knowledge, when a person has

more knowledge, the person generally thinks better and is more effective at understanding the environment and functioning in it. When computers have more knowledge, they tend to bog down in searching through the additional facts and rules that have to be considered. I believe that this difference is due to something about the way people organize knowledge for efficient use, and I’d like to be able to get computers to do the same thing.

To give you an example of the kind of thing I have in mind, consider the following example. At one time I was trying to remember the name of a flute piece that my wife used to play, and the thing that kept coming to mind was *Fahrenheit 451*, which I knew was wrong. *Fahrenheit 451* is a book about censorship and book burning, and the title is the temperature at which paper burns. Eventually I remembered that the piece I was trying to recall was called *Density 21.5*. Density 21.5 is the density of platinum, and the piece was written in honor of the first platinum flute. Both of these titles use essentially the same device, and this device is so unique that the first time we see it, we are likely to form a concept something like “a creative work whose title is the statement of the value of a physical property that has a semantic relationship to the work.” When the second occurrence of this device was encountered, I apparently recognized the similarity at this abstract level, and the two facts were thus related to each other. When I later tried to recall the musical piece, something about the way my mind works was causing the abstractly similar book title to be recalled out of all of the millions of things that I know. This particular abstract generalization was triggered by my much more specific recall attempt. It happened effortlessly and with no conscious awareness of the process. This kind of retrieval is the capability that lets us recognize quickly what kind of situation we are facing, what to expect in that situation, what to do about it, and what other similar situations we have encountered. These capabilities are key to our successful adaptation to the world in which we live and are part of the essence of intelligence.

My goal is to discover a kind of “mental” organization of knowledge that would enable a computer to do a similar kind of retrieval from large knowledge bases (Woods 1986a). This is key to developing scalable knowledge-based systems. Without it, systems are limited in the amount of knowledge that they can handle, and projects like the semantic web will be unable to deal with truly web-scale knowledge.

In passing, it is worth noting here that many “knowledge-based systems” are actually what I

have called “ignorance-based systems” because they rely on knowing only what they need to know to solve the problem and on not knowing the many potentially distracting things that could lead them down blind alleys and increase the combinatoric search for a solution. As a simple example, when writing grammars to parse sentences, it is much easier if the parser can assume that “I” is always a personal pronoun and “be,” “am,” and “are” are verbs. It becomes more difficult if the parser’s dictionary also knows that “I” is the chemical element symbol for iodine, the roman numeral for one, the mathematical symbol for the square root of minus one, the name of the ninth letter of the alphabet, and even an Asian last name. These six choices can be multiplied by two for each occurrence of “be,” “am,” and “are” if the system also knows that “Be” is the chemical element symbol for Beryllium, that “am” is an abbreviation for “morning,” and that an “are” is a measure of land area equal to one hundred square meters. If a reasoning system can handle this kind of additional and potentially distracting knowledge, then it is truly robust and scalable; otherwise, it can be thrown into combinatoric distraction by the addition of more knowledge.

## History

My concern with semantics and meaning began in a seminar at Harvard University in which my eventual thesis advisor, Susumu Kuno, posed the problem of designing a natural language question-answering system to query a database. After some thought, it seemed to me that if one were going to get a computer to answer questions, then one had better know something about meaning, so I turned to the library and read a lot of philosophy: Tarski, Wittgenstein, Carnap, Quine, Church, and others. None of this told me what meaning was, however. The best I could find was essentially an operational test for whether one knew the meaning:

To know the truth conditions of a sentence is to know what is asserted by it—in usual terms, its “meaning” – *Rudolf Carnap*

But truth conditions are a very abstract thing and, in particular, an infinite thing—a mapping of all possible worlds into the values true or false. What could a computer use (or a person for that matter) that would meet this operational test? What could be stored in a finite memory that would encode the truth conditions for an infinite set of possible worlds?

The only thing I knew that would meet this test was some encoding of a procedure—a Tur-

ing machine, a production system, a recursive function definition—essentially a computer program for computing the truth values from the possible world itself. This idea, which I introduced in my thesis (Woods 1967) and called “procedural semantics” (Woods 1968), not only turned out to be a powerful way to build a practical question-answering system, but also solved a classical problem in the philosophy of language—how the meaning of a proposition could be connected to the actual physical world. In the procedural semantics framework, the meaning of a proposition (that is, the procedure expressing its truth conditions) could be embodied (stored) in a physical machine that used the procedure to operate on the physical world to compute the truth values. For example, a question about the inventory of bolts in a warehouse could be answered by a robotic system that actually went to the appropriate bin in the warehouse and counted the bolts. This pure idea turned out to be somewhat oversimplified, but a more refined version, using a notion of abstract procedures, which I presented in two essays on procedural semantics (Woods 1981 and Woods 1986b), turns out to come closer to being a viable theory of meaning. This more refined theory distinguishes between abstract procedures (possibly not even executable) that determine the meaning of a term and other (possibly fallible) recognition procedures that are ordinarily used to infer when the term is true or false.

The term *procedural semantics* has since been used by a variety of people with various meanings, often associated with representations that look like computer programs. This is opposed to taking abstract procedures as the semantic foundation for representations that could look like logical expressions or like network representations. I will have more to say about procedural semantics in a subsequent section of this article. For more information on the history of the idea and the so-called procedural versus declarative controversy, see Woods (1987b) and the preface to the published version of my thesis (Woods 1979).

This idea of procedural semantics became the basis of my Ph.D. thesis, “Semantics for a Question Answering System,” which I finished in 1967 and was eventually published in 1979. In that thesis, I presented a methodology for mapping a parse tree for a natural language question into a computer program for computing its answer—essentially a natural language compiler. I argued for the separation of the parsing and semantic interpretation machinery from the back-end database and reasoning machinery by means of a general notation for

expressing these procedures (what is now called a meaning-representation language). Prior to this, question-answering systems had usually transformed questions into the notation in which the data was represented, and questions were answered by pattern matching, equation solving, logical deduction, or database retrieval, depending on the representation chosen. For example, Robert Simmons (1965) describes a system that parsed a question into a tree structure, replacing the question word with a variable. This system answered questions by matching this structure against the parse trees for sentences from an encyclopedia and reported the value matched against the variable as the answer. While this worked well for many examples, in response to the question "What do worms eat?" this system found "Worms eat their way through the ground." In another example, Daniel Bobrow (1964) transformed sentences from high school algebra word problems into equations in algebra and answered questions by solving algebraic equations.

Incidentally, I developed the formalism of augmented transition network (ATN) grammars (Woods 1970 and Woods 1987a) in order to have a practical way of parsing English sentences into the kinds of parse trees that I needed as input to my semantic interpreter. At the time, Chomsky's theory of transformational grammar was a hot topic, Haj Ross and George Lakoff were furiously working out details of the grammar of English within this framework (and forcing extensions of the framework in the process), and Stanley Petrick was working on a parsing system for transformational grammars at MITRE. Sheila Greibach, Susumu Kuno, and Tony Oettinger had recently developed the Harvard Predictive Analyzer, a large-scale context-free grammar for English, and had explored the richness (and ambiguity) of natural language syntax by parsing such things as nuclear test ban treaties. (This was in the context of a larger effort in machine translation.) In my doctoral thesis, I had cited this work as a justification that English sentences could be parsed into tree structures, and the thesis focused on the task of assigning semantic interpretations to such trees. Thus, when it came time to actually implement such a system, I needed a practical way to transform natural English sentences into parse trees, and my notion of ATN grammars was invented in response to that need.

In 1970, the theory and practice of both the ATN grammar formalism and the method of procedural semantics were put to the test in the Lunar Sciences Natural Language Information

System (LUNAR), which several colleagues and I developed at Bolt Beranek and Newman (BBN) for the NASA Manned Spacecraft Center (Woods et al. 1972). This system answered questions about the chemical composition and other experimental analyses of the lunar rocks returned from the *Apollo 11* moon mission. It was the first question-answering system to report empirical results from a test of the system with real users (scientists attending the Second Annual Lunar Science Conference). One of the features of this system was a powerful framework of generalized quantifiers for handling the kinds of quantification expressed in natural English questions (Woods 1978).

In the midst of all this linguistic activity, I noticed that while the philosophers thought of semantics in terms of truth conditions, the linguists had different notions. Their criterion for a meaning representation was the ability to produce different representations for the different "readings" they could perceive in an ambiguous sentence. For example, "Time flies like an arrow," has a reading in which flies of a certain kind ("time flies") are fond of an arrow (analogous to Groucho Marx's rejoinder: "but fruit flies like a banana"). The "meaning" representations of the linguists were essentially parse trees, and when they talked about "semantic conditions," the mechanisms they had in mind were the same kind of thing that parsers used for syntactic conditions. For example, "the knife shoots bullets" was considered ungrammatical because of a violation of a "semantic" condition requiring the subject of "shoot" to have a "semantic" feature "+agent." There was nothing in the envisioned machinery that had anything to do with knowing what a knife was or what shooting involved (leaving aside the issue of whether ungrammaticality is the right concept for expressing what is strange about this sentence).

When it came to artificial intelligence researchers talking about representing knowledge in semantic networks, the lack of appreciation of what *semantics* should mean was even worse. In fact, there was generally nothing really semantic in *semantic networks*, and I felt the term itself was a misnomer. It was in response to these observations that I wrote the paper "What's in a Link."

The essential content of the "What's in a Link" paper was developed at a brainstorming workshop held at a National Academy of Sciences conference center in Quisset, Massachusetts, on Cape Cod. I was attempting to tell people about some of the mistakes and inconsistencies I saw people making when talking about semantic networks for representing

knowledge. These mistakes, I felt, stemmed from a lack of understanding of what semantics meant and what it would take for a machine to correctly interpret such representations. I attempted to describe some of the issues that needed to be addressed and possible ways to address them.

After a subsequent conference at Pajaro Dunes, in California, Danny Bobrow encouraged me to finish the paper and provided access to the facilities at Xerox PARC as a way to help make that happen. The final version was produced using the Bravo text editor on an Alto computer at Xerox PARC. (This was the computer that pioneered bitmap graphics, overlapping windows, WYSIWYG editing, and laser printing.)

## Semantics and Meaning

The meaning of *meaning* and how to deal with meaning in formal and natural systems has been one of the great mysteries of intelligence—artificial or otherwise. It has been an issue from the earliest days of philosophy and logic, and it has become an engineering issue with the advent of computerized question-answering systems, information retrieval systems, machine translation, speech understanding, intelligent agents, and other applications of natural language processing, knowledge representation, and artificial intelligence in general. When I first started working on question answering, I was immediately confronted with the fact that semantics in classical logic dealt only with logical propositions (things that are true or false), while questions and imperative requests are not things that have truth values. What kind of thing is the meaning of a question? What is the relationship between a question and an answer? What is the meaning of an imperative request? What is meaning anyway?

As I mentioned previously, early work on question answering had involved first choosing a representational convention and then transforming a question into the same notation so that it could be processed with a suitable algorithm. One of the limitations of this approach was that it required analyzing the questions into the same notation and conventions as were used in representing the data, so that the structures of the questions and the data were compatible. Question-answering systems that took this approach could not be combined unless all of the content was analyzed using the same representational conventions. Moreover, the complicated work of understanding the syntactic structure of English questions had to be redone for each system

in terms of the representational conventions chosen. I was interested in what it would take to have a general notion of meaning that would be independent of the representational conventions into which the content was analyzed and that would allow the integrated use of data that was represented in different ways. Moreover, I wanted to develop a general-purpose English grammar and parser and a general-purpose semantic interpreter that could be used with sentences from any natural language and would transform them into a common, fully expressive meaning-representation language. In order to do this, it seemed that I needed a good theory of meaning that could be independent of particular data representations and particular natural languages.

Conceptually, question answering can be thought of as involving several subtasks, including parsing, semantic interpretation, and some combination of retrieval, inference, and computation. Of these, the most mysterious phase was semantic interpretation, which seemed to involve meaning. Semantics can be defined rather crisply as the relationship between terms or expressions and the things that they denote or mean. Thus the term *semantics* is relatively well defined in terms of the concept of meaning, but the meaning of *meaning* is much less clear. In the “What’s in a Link” paper, I pointed out the different uses of this term by the people I characterized as “linguists” and the people I characterized as “philosophers,” and I pointed out that there was still a missing element necessary to map the possible worlds of the philosophers into something that a computer (or a person) could store and manipulate. The solution I proposed for filling this gap was my theory of procedural semantics, which could not only express truth conditions as abstract procedures but could also express the meanings of questions and imperative requests and of sensory perception and motor action.

## Procedural Semantics

The idea of procedural semantics is that the semantics of natural language sentences can be characterized in a formalism whose meanings are defined by abstract procedures that a computer (or a person) can either execute or reason about. In this theory the meaning of a noun is a procedure for recognizing or generating instances, the meaning of a proposition is a procedure for determining if it is true or false, and the meaning of an action is the ability to do the action or to tell if it has been done.

This theory can be thought of either as an

```
(FOR EVERY X5 / (SEQ TYPECS) : T ;
  (PRINTOUT (AVGCOMP X5 (QUOTE OVERALL) (QUOTE AL2O3))))
```

Figure 1. The Semantic Interpretation of the Question, "What Is the Average Concentration of Aluminum in Each Breccia."

From the LUNAR question-answering system (1971).

alternative to the standard Tarskian semantics for formal systems or as an extension of it. The idea is that the computational primitives, consisting of represented symbols, the ordered pair, the assignment of values to variables, the addition and subtraction of integers, conditional branching, iteration, and the subroutine call, together with sensorimotor operators that interact with a real world, constitute a stronger (and more well-understood) foundation on which to build a theory of meaning than do set theory and the logical operations of universal and existential quantification over an all-inclusive infinite universe (which have their own paradoxes and incompleteness issues). For more about the limitations of classical logic alone as a basis for meaning, see Woods (1987c).

Adopting a procedural semantics foundation allows for a definition of the standard logical operators as well as extensions of them to deal with generalized quantifiers and with questions and imperative requests. Building on computational primitives seemed to me to be at least as well understood, and more grounded, than building on universal and existential quantification over an all-inclusive infinite universe (and then having to make some kind of extension to handle the meanings of questions and imperatives). Moreover, because procedural specifications can be installed in a machine, where they can be physically executed, they can interact with sensors like keyboards and cameras and with output devices like printers and manipulators, so that this approach allows meanings that actually interact with a physical world, something that no previous theory of meaning had been able to achieve.

In the case of question answering, procedural semantics allows one to decouple the parsing and semantic interpretation of English sentences from the details of the storage and representational conventions of the information in the database. The procedural semantics approach allows a computer to understand, in a single, uniform way, the meanings of conditions to be tested, questions to be answered, and actions to be carried out. Moreover, it permits a general-purpose system for language understanding to be used with different databases, and even combinations of databases, that may have

different representational conventions and different data structures. It also allows questions to be answered by results that are computed from the data in the database without being explicitly stored.

This can be illustrated by an example from the LUNAR system. The semantic interpretation of the question, "What is the average concentration of aluminum in each breccia," is given in figure 1.

This semantic interpretation can be read as follows: "For every x5 in the class TYPECS such that the universally true condition T is true, print out the value computed by the averaging function AVGCOMP for the sample x5 for the overall concentration of Al2O3," where TYPECS is the name used in the database for "Type-C rocks" (that's how breccias are encoded in the database) and Al2O3 is the chemical name for aluminum oxide (which is how aluminum concentrations are represented in the database). Note that answering this question involves computing an average that was not explicitly stored in the database, and that the grounding for the semantics of terms such as *aluminum* and *breccia* is in a database whose structure and content and encoding conventions were previously and independently defined.

The quantifier notation in LUNAR used special symbols, "/" ":" and ";" as illustrated in figure 1, to separate different elements of a quantification. For improved readability, for the rest of this article, I will use the mnemonic keywords, "in," "when," and either "do" or "thereis," respectively, in place of these special symbols (";" becomes either "do" or "thereis" depending on whether the quantified expression is a command or a predicate).

## Meaning Representation Language

LUNAR's meaning representation language (MRL) is an extension of the Predicate Calculus with generalized quantifiers and imperative operators. Some examples of the schemata for these quantifiers and operators are shown in figure 2.

For example, the first schema in figure 2 can be read "For <quant> <vbl> in the class <class> such that <condition> is true, do <command>." This is the schema for a quantified command. The second schema can be read: "For <quant> <vbl> in the class <class> such that the first <condition> is true, the second <condition> is also true." This is the schema for a quantified proposition. The quantifiers <quant> in these schemata include not only

the traditional universal and existential quantifiers EVERY and SOME, but also nontraditional quantifiers like THE and (MORETHAN <number>) and a generic quantifier GEN, which corresponds to the use in English of undetermined noun phrases with a plural noun (for example, birds have wings). The paradigm can accommodate numerical quantifiers as exotic as “an even number of” or “a prime number of” and all sorts of nonstandard quantifiers such as “few” and “most.” The paradigm can also handle modal wrappers like “typically,” “often,” “rarely,” and “unless-contradicted.”

The schema (TEST <condition>) is a command to test a condition and print out Yes or No according to the result, and (PRINTOUT <designator>) is a command to print out a name or description of the referent of the specified designator.

LUNAR’s meaning representation language uses typed quantifiers, with types that can be functionally specified (that is, the types can take parameters). The type classes over which quantified variables could range were defined by a kind of generator (I called them enumeration functions) which could enumerate the elements of the class. These enumeration functions could take arguments (the parameters of the type), so that the class generated could be functionally determined. This allowed a uniform treatment of nouns in English that named classes and nouns that took arguments (usually expressed as prepositional phrases) like the departure time of a flight from a city. The notation also allows additional restrictions on the range of quantification to be specified by predicate filters. This allows a uniform treatment of different quantifiers and different kinds of nouns and modifiers in English noun phrases. For example, “Some tall men play basketball” has the interpretation:

```
(FOR SOME X in MAN when (TALL X) thereis
(PLAY X BASKETBALL))
```

and “All long flights to Boston serve meals” has the interpretation:

```
(FOR EVERY X in (FLIGHT-TO BOSTON) when
(LONG X) thereis (SERVE-MEAL X))
```

If we try mapping English directly to classical logic, we need different treatments for noun phrases with universal versus existential quantifiers. For example, for “Some tall men play basketball,” the three predicates (MAN X), (TALL X), and (PLAY X BASKETBALL) are all conjoined under the existential quantifier:

```
(for some x) {x is a man and x is tall and x plays
basketball}.
```

For “All tall men play basketball,” the first two

```
(FOR <quant> <vbl> in <class> when <condition> do <command>)
(FOR <quant> <vbl> in <class> when <condition> thereis <condition>)
(TEST <condition>)
(PRINTOUT <designator>)
```

Figure 2. Quantificational and Imperative Schemata Used in the LUNAR Question-Answering System.

conditions would be conjoined as the antecedent of an implication whose consequent is the third condition:

```
(for all x) {if x is a man and x is tall, then x plays
basketball}.
```

For nonstandard quantifiers, such as “the” or “many” or “most,” mapping to classical logic is different still, but all fall uniformly into LUNAR’s MRL paradigm.

## Reasoning with Procedural Semantics

The procedural semantics framework allows procedural interpretations to be treated in two ways: In the simplest way, the semantic interpretation is simply executed as a program. However, in some cases, the system will take the interpretation as an object to be reasoned about and possibly modified. For example, in the following case from an airline flight schedules application:

```
(FOR EVERY X in FLIGHT when (CONNECT X
BOSTON CHICAGO) do (PRINTOUT X))
```

the system discovers that it can get the same result more efficiently by using a functionally specified range of quantification that uses a database table to enumerate only the flights that go to Chicago, resulting in:

```
(FOR EVERY X in (FLIGHT-TO CHICAGO)
when (CONNECT X BOSTON CHICAGO)
do (PRINTOUT X))
```

This is an example of what I have called “smart quantifiers,” quantifiers that apply reasoning to their range of quantification and any specified filters on that range to see if there are more efficient ways to enumerate the same effective range.

There are many other reasons why a system might want to reason about the interpretation of a request before acting on it: it may not be appropriate to take literally, or there might be some additional helpful actions to take, or the request might be estimated to be unduly expensive, or the user may not be authorized to do what is being requested, or the system

may have reasons not to do the thing that is requested, or it may be more efficient to reorder the quantifiers, or... So, we would like semantic representations to be useful both for execution as procedures and for mechanical reasoning.

## Requirements for Semantic Representation

Semantics is used in natural language processing not only as an end result, but also in order to choose between alternative syntactic parses and alternative senses of words. For example, knowledge is often required to choose the scope of modifiers when parsing English sentences, as in the question "Does American have a flight from an East coast city to Chicago?" which my thesis system interpreted with the phrase "to Chicago" modifying "flight" rather than "city" (even though "city" is closer) because it didn't have any rules for cities being modified by destinations, but it did have a rule that knew what it meant for a destination phrase to modify "flight." In the case of speech understanding, semantic interpretation and background knowledge can be required even to determine what words were heard. For example, in the BBN HWIM Speech Understanding System (Woods et al. 1976), the sentence "Show me Bill's trips to Washington" was misheard as "Show me Bell's trips to Washington" in the context of a travel planning system that knew travel plans for a group of people that included Bill Woods and Alan Bell. There is a minimal difference of one phoneme between these two sentences (one letter, in the written orthography) and only one feature difference between the two vowels in question. The acoustic scores of the two hypotheses were virtually identical and the correct choice happened to come second. However, the system could easily have resolved the choice by using the interpretation to check the trip database to learn that Bill Woods was scheduled to go to Washington, while Alan Bell was not.

Thus, we need a system that can organize and use large amounts of world knowledge and facilitate the efficient associative access to that knowledge during the analysis of sentences. My experiences in a variety of natural language applications have convinced me that understanding and using knowledge is the bottleneck in both speech and natural language processing and that finding and retrieving the relevant pieces of knowledge that apply to a problem at hand, from among all of the knowledge in a large knowledge base, is a significant issue. We need a representation system to sat-

isfy at least two requirements: (1) it should be expressively adequate to represent all of the necessary elements of natural language questions, commands, assertions, conditions, and designators, and (2) it should be structured to support semantic interpretation, retrieval, inference, and perception.

For further discussion of requirements for a knowledge representation system, see Woods (1986a and 1987c) and Davis, Schrobe, and Szolovits (1993). In order to attain human-level reasoning, I believe that it will be necessary to combine the best features of two traditions: (1) logical reasoning, which is rigorous and formal, but often counterintuitive, and which has algorithms that match expressions, substitute values for variables, and invoke rules, and (2) associative networks, which are associative and intuitive, but typically informal. However, they support efficient algorithms that follow paths through links to draw conclusions.

We need the associativity of link-based representations in order to exploit efficient path-following algorithms, but we also need representations with a clean and well-understood semantics. To that end, I have been interested in the things people do with link-based representations and seeking conventions for making explicit the semantics of the links in such representations.

## Semantic Networks

I had the privilege to be working at BBN at the time when Ross Quillian, Allan Collins, and Jaime R. Carbonell were working there on a variety of projects with semantic networks, one of which was Jaime's mixed-initiative tutoring system called SCHOLAR (Carbonell 1970). This was a computer-aided instruction system to tutor students about South American Geography, and it used a semantic network as its knowledge base. Jaime worked with Allan Collins on SCHOLAR, and the pair of them had great insights into how people use knowledge and how knowledge could be represented and used in a computer system for educating people. (I should explain that the Jaime R. Carbonell I refer to here is the father of the current Carnegie Mellon University professor.)

SCHOLAR used its semantic network to represent the knowledge that the tutor had to teach, and it would annotate this network with records of what the student now knew as it introduced material and tested the student's knowledge. It could use these annotations to assess how much a student knew about a given concept and to decide what material to introduce next. It could also use the network to

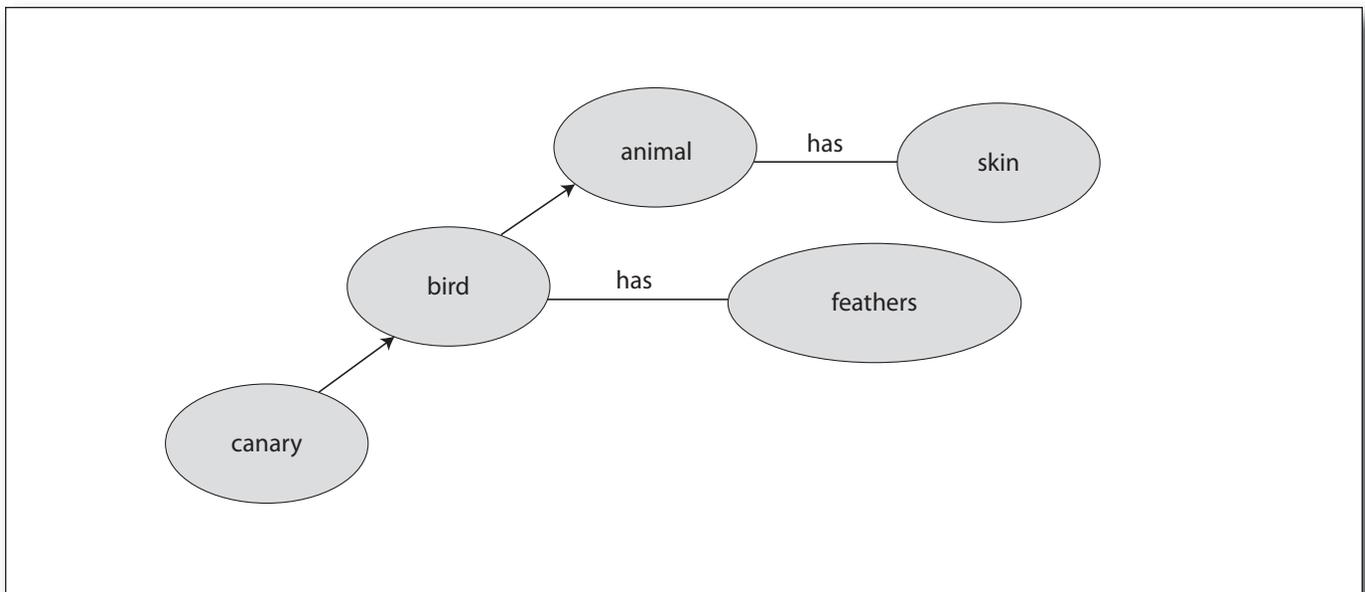


Figure 3. Illustration of Semantic Inheritance.

assess how much it knew about a given topic and could apply this self-knowledge in answering questions with what Collins called the lack-of-knowledge-principle, a sophisticated kind of local closed-world assumption (Collins et al. 1975). If you asked it whether Bolivia exported steel and it didn't know that Bolivia exported steel, but it knew enough about Bolivia and its exports to be confident that it would know this if it were true, then SCHOLAR would answer confidently "No." If it didn't know enough to be confident in a negative answer, it would answer "I don't know."

Unfortunately, Jaime died suddenly and unexpectedly in the middle of this work and the project ended.

### Inheritance in Semantic Networks

Perhaps the first and simplest algorithm for the use of links in semantic networks was the concept of inheritance, now a common feature of knowledge representation systems and object-oriented programming. In the early days, Quillian and Collins were seeking to understand whether this kind of representation played a role in human reasoning (Collins and Quillian 1969). Figure 3 illustrates an example from one early experiment that they carried out to investigate the psychological reality of a path-following algorithm in a semantic net. In this experiment, Collins, who is a psychologist, presented people with a variety of statements and measured their reaction time to decide whether

the statement was true or false. In each case, Collins and Quillian had made assumptions about the background knowledge people would use to answer the question, how that knowledge would be represented in a semantic net, and how that representation would affect reaction time. Specifically, they predicted that it would take longer to answer when the attribute was inherited from a higher category, so that in the example illustrated, it should take less time to decide whether a canary has feathers than to decide whether a canary has skin. Their predictions were validated in the experiments, but the results were not conclusive because other explanations could have predicted the same results.

In other experiments, Quillian would test his networks by running an algorithm to describe the relationship between arbitrary pairs of concepts. This would be found by a "spreading activation" algorithm that searched for the shortest path between nodes in the semantic network. This work presaged later work by Rummelhart, Norman, and others on psychological aspects of network representations and on various algorithms for learning in such networks.

These algorithms were really cool and many of them had a plausible psychological reality as models of how people do associative reasoning. However, all of these methodologies were very informal in terms of the semantics of what the links in the networks represented.

Many other people were using network representations at that time, and there was (and perhaps still is) an astounding range of opinion

about the meanings of such representations or whether meaning even matters. Generally, people working in this area were more interested in the things that an algorithm could do with the representations than with what the representations signified, and they would casually use similar representations to mean different things in different experiments. Most of the people who did say something about the meanings of their representations seemed to me to be well off the mark. Basically, this is what led me to write the “What’s in a Link” paper. Here are some of the inadequate notions of meaning that some people espoused, many of which may still be in people’s minds today:

1. The meaning of a concept is the totality of the concepts connected to it.
2. The meaning of a concept is whatever a system does with it.
3. There’s no difference between syntax and semantics.
4. Semantics is in the eye of the beholder, because the meaning comes from the names of the nodes.

In the first part of my “What’s in a Link” paper, I attempted to clarify what a semantic account would entail. I described the different perspectives on semantics held by what I have already characterized as the linguist versus the philosopher, which, even combined, still fell short of answering my needs for something that could be stored and used by a computer. I proposed procedural semantics to fill the remaining gap.

The second part of the paper addressed the various things that a link could be used to mean and the many ways in which people were not clear about what their representations meant. When I looked for the semantics of links that I saw people using, I found many different things that were being represented or would need to be represented as links:

attributes and values  
 attributes and value-predicates  
 relations and objects-of-relations  
 functions and arguments  
 actions and objects  
 roles and constituents  
 roles and constituent-restrictions

The most prototypical case is using links for attribute-value relationships, as in:

```
John
  height      6 feet
  haircolor   brown
  occupation  scientist
```

But what if all you can say about a value is that it satisfies some predicate, as in:

```
John
  height(greaterthan 6 feet)
```

Or worse, what if you want to compare the values of two attributes, as in “John’s height is greater than Sue’s”:

```
(height John)
  greater (height Sue)
```

Sometimes people used links to assert facts, as in:

```
John
  hit   Mary
Mary
  hit*  John
```

where the “hit\*” link here is the reverse of the forward “hit” link.

In some cases, both types of links branch from the same node, with nothing to indicate that they have different kinds of interpretation, as in:

```
John
  height 6 feet
  hit     Mary
```

Links were sometimes used to build descriptions using case grammar constituent roles, as in:

```
sell
  agent      John
  recipient  Mary
  patient    book
```

But sometimes the same representation was used to state the value restrictions on the case roles:

```
sell
  agent      person
  recipient  person
  patient    thing
```

I identified a distinction between assertional links, which assert facts, versus structural links, that merely build up parts of a description of something about which something else may be asserted. Sometimes, the mere existence of a description of a relationship in a semantic network corresponds to the assertion that the relationship holds, but if so, then there is no way to represent such a description as the object of a denial. Often it was not clear whether a representation was intended as an assertion or as a description of something. I cited the following example, which either represents a black telephone or asserts that telephones are black, depending on which kind of interpretation is given to the links:

```
N12368
  superc      telephone
  color       black
```

All of these things were represented by links in various systems and examples. These different kinds of link relationships have very different semantics, and any algorithm that operates on these representations needs to understand these semantics in order to function as intend-

ed. Shifting from a convention in which links point to values, to a convention in which links point to predicates that are true of values, would require a systematic change to the representations for everything and a corresponding adjustment to the algorithms that operate on the representation. The distinction between links that make assertions and links that build parts of descriptions (about which assertions are then made) makes things even more complicated. Ideally, each link should be associated somehow with a characterization of its semantics (for example, whether it is a pointer to an individual value, a pointer to a predicate that applies to the value, or a pointer to an abstract description about which assertions can be made). If this is not the same for all links, then each link needs to be accompanied with some source of information about how it is to be interpreted.

I concluded the article with examples showing the need to represent at least the functional equivalent of LUNAR's generalized quantification, and I demonstrated the need to represent propositions without commitment to their truth and to represent descriptions of individuals without commitment to their existence.

## Towards an Internal Notation for Thought

After "What's in a Link" was published, I began a new DARPA project at BBN aimed at representing knowledge and at addressing the issues raised earlier. This was the KL-ONE project that Ron Brachman and I and many other colleagues worked on (Brachman and Schmolze 1985). KL-ONE pioneered a number of concepts that are now well known, including the concept of automatic concept classification, which can be done only if the semantics of the notation are well understood. The project started out with the somewhat nebulous goal of providing the representational framework for fully general knowledge representation and reasoning on large-scale knowledge bases. This included uncovering and attempting to understand subtle issues of representational semantics and exploring the consequences of intuitions about the role of knowledge in reasoning and in language understanding that we were still trying to work out.

A central starting point of this project was Ron Brachman's Ph.D. thesis (Brachman 1977) on structured inheritance networks. Ron was one of my Ph.D. students at Harvard and also worked at BBN. His thesis made distinctions between primitive concepts as compared to defined concepts and distinguished concepts, roles, number restrictions, and value restric-

tions. As a result of the relatively crisp semantics of these concepts, we were able to create algorithms that could automatically decide where new concepts belonged in a generalization hierarchy. This was discovered in the following way.

In order to support the acquisition of knowledge, I wrote an English-like language for entering knowledge into the system and wrote an algorithm to decide where to add new knowledge into an existing knowledge taxonomy. This algorithm consisted of two parts, a most-specific subsumer (MSS) algorithm that could find the most specific subsumers of the new concept (that is, the most specific concepts in the taxonomy that subsumed the new concept) and a most-general subsumee (MGS algorithm that could find the most general subsumees of the new concept (that is, the most general concepts in the taxonomy that were subsumed by the new concept). The new concept could then be added to the taxonomy, linked directly under its most specific subsumers and linked to from below by its most general subsumees. This was the first of what came to be called classification algorithms. Jim Schmolze and others worked on subsequent classification algorithms that were more efficient.

Technically, the structure referred to previously as a generalization hierarchy was not strictly a hierarchy, but was rather much closer to an upper semilattice, since a concept could in general have multiple parents (that is, most specific subsumers). Moreover, it is not necessarily a strict semilattice either, since it could in principle allow multiple least upper bounds and greatest lower bounds. Hereafter, I will refer to this kind of structure as a conceptual taxonomy.

The original goals that I had for the KL-ONE project were quite ambitious. I wanted an organizing structure for all of the knowledge of a reasoning system and an efficient associative access mechanism to get at that knowledge, and I wanted the representation to have complete expressivity and support for high-level perception and automatic classification and to support inheritance of attributes and rules. We pursued a variety of ideas in all of these dimensions. Conferences were organized and a variety of people from many organizations shared ideas on how to deal with a wide variety of subtle representation problems. This was an exciting time, and these conferences were the origins of the series of Knowledge Representation and Reasoning conferences.

Eventually the complexity of KL-ONE's many goals led Ron Brachman and Hector

Levesque to focus on some specific problems relating to expressivity, and they discovered the first expressivity/complexity trade-off (Brachman and Levesque 1987). (This is my perspective—Ron and Hector might describe it differently from their point of view.) After this, there were many theoretical complexity results and many new systems in what came to be called the KL-ONE family (Woods and Schmolze 1992), eventually giving rise to a new field that changed its name periodically, but seems to have settled on description logic (Brachman and Levesque 2004).

## Extensional Subsumption

A common theme of this post-KL-ONE theory work was a move away from thinking of concepts as structured objects in favor of a “declarative semantics” that focused on the subsumption relationship in terms of a model-theoretic semantics. In this view, which I call extensional subsumption, one concept subsumes another if by virtue of the meanings of the concepts, any instance of the subsumed concept must necessarily be an instance of the subsuming concept in a model theoretic sense. That is, the extension of the subsuming concept (the set of all objects that are its instances) contains the extension of the subsumed concept in any possible world.

This turned out to be a challenging criterion to meet, and most of the complexity results yielded algorithms that were computationally intractable in the worst case. Completeness with respect to this criterion is difficult to achieve, and full expressivity leads to the complexity of predicate calculus deduction in general.

## Understanding Subsumption and Taxonomy

While most of this flurry of theoretical activity was going on, I was involved in a couple of start-up companies and watching all this from the sideline. When I came back to these issues, I concluded that the declarative semantics approach had lost something important by eliminating all of the intuitions for how the structure of links can support efficient algorithms and that the extensional subsumption criterion was also limited in its ability to express subtle meanings. In 1985, I began to revisit the original goals of KL-ONE in light of where the field had gotten. I wanted a representational system that would be an efficient and principled methodology for organizing knowledge, and I came to focus on a different criterion for subsumption that I called an *intensional*, rather

than extensional, subsumption criterion. The result was my 1991 paper, “Understanding Subsumption and Taxonomy” (Woods 1991).

## Intensional Subsumption

The idea of intensions is an old one in the philosophy of language and is typically illustrated by the distinction between the concepts of the morning star and the evening star. Before the discovery of astronomy these were thought to be two different stars, and only subsequently was it discovered that both descriptions referred to the planet Venus. The case is made that if the two descriptions meant the same thing, then it wouldn’t have taken an astronomical discovery to reveal their identity. Therefore, the meanings of these descriptions must be something other than their common extension (that is, Venus).

This example demonstrates the need for distinct abstract entities (intensions) that stand between these literal phrases and their shared referent. Intuitively, the morning star refers to the last star that is visible in the morning, before the sun comes up, while the evening star refers to the first star that is visible in the evening. Notice that both of these are essentially procedural descriptions and thus a procedural semantics interpretation of their meanings could provide a suitable intensional abstraction. That is, one can take the meanings (intensions) of these concepts to be the abstract procedures for recognizing them.

Carnap thought that since this identity was a contingent fact, not a logical one, he could use the concept of logical equivalence to create intensional abstractions. However, I have discovered that the concept of a triangle is an analogous example, in which the identity is a necessary consequence of the meanings of the terms, and thus Carnap’s “L-equivalence” solution will not suffice. That is, the definitions “polygon with three sides” and “polygon with three angles” are logically equivalent, but it is a common exercise in geometry class to prove this. If these two descriptions meant the same thing, then it would not have been necessary to prove the equivalence. In fact, it would not even be possible to state the theorem, since any attempt to state it would be equivalent to stating the tautology, “an  $x$  is an  $x$ ,” where  $x$  could be either one of the descriptions. (I will say more about this later.)

The idea of intensional subsumption is this: Since the meanings of concepts need to be intensions rather than extensions, the criterion for subsumption can’t be based on extensional set inclusion but must be based on

something that takes the intensional meaning of the concept into account. In particular, this means that different concepts with the same extension can remain distinct, without the classifier wanting to merge them into a single concept.

From my perspective, sameness of intension is determined by a definition of intensional equivalence that specifies when two descriptions are to be considered intensionally the same. For example, different orders of listing the roles of a structured concept would be considered equivalent, but logically equivalent characterizations involving completely different constituents might not. What should count as equivalence of intension is as much a psychological as a logical issue. If people can argue that there are potentially different meanings of two representations, then intensional equivalence of those representations can be ruled out. From the perspective of abstract procedures in procedural semantics, there can be different notions of intensional equivalence, each defined by its own equivalence relation. For example, the order of adjectives in an English noun phrase might not make a difference in most cases. However, since there are cases like “little old lady” versus “old little lady” (that is, a sagacious little girl), where the order of modifiers does make a difference, one may need to provide a notation or a convention in which these two noun phrases have different meanings.

In addition to dealing with intensional meanings, I was interested in defining a notion of intensional subsumption that would make the subsumption test something that could be verified by inspection, without needing subtle proof. If an extensional subsumption required subtle proof, then that would have to be done by a separate system, after which that subsumption could be asserted as an axiom, justified by that separate proof, but it would not be the job of the classifier (that is, the subsumption algorithm) to find it. Finally, I wanted the overall classification of a concept into a large taxonomy to be efficient (preferably sublinear in the size of the overall taxonomy). An important aspect of intensional subsumption is that it not only allows for the representation and handling of intensional concepts, but it also escapes some of the complexity limitations of extensional subsumption.

### Separating Necessity and Sufficiency

One of the things I discovered when investigating this notion of intensional subsumption

is that one can gain additional expressive power by allowing a concept to have different necessary and sufficient conditions. I thus allowed three kinds of definitional statements and allowed a concept to have more than one such definition:

<concept> [def-if] <condition>  
*specifies a sufficient condition*

<concept> [def-only-if] <condition>  
*specifies a necessary condition*

<concept> [def-iff] <condition>  
*specifies a condition that is both*

This allows us to represent partially defined concepts for dealing with natural kinds and overdefined concepts for dealing with certain kinds of abstraction (for example, “triangle”).

### Natural Kinds and Gap Predicates

Natural kinds like “tiger” and “chair” are typically thought of as undefinable, since any attempt to define them seems to turn up short. However, they can often be given partial definitions by specifying some necessary conditions and separately defining a different and more stringent set of sufficient conditions. In between these two conditions is a gap in which things are neither in or out. This is not possible in classical first-order-logic axiomatizations, but is not a problem with a procedural semantic interpretation. I have argued elsewhere that decisions about whether a hypothetical example that falls in this gap should be judged an instance of a natural kind or not is a different kind of decision than deciding whether an instance satisfies a predicate. Rather, it is a decision whether the utility of the concept in question will be enhanced or damaged by extending the predicate to include the new example (for example, will all of the learned rules and axioms that use that concept still be valid and useful, or will some of them be violated).

### Multiple Definitions and Abstract Concepts

Multiple definitions can be used to define abstract concepts like triangle, which as we mentioned has two alternative definitions, polygon with three sides and polygon with three angles. These two concepts are extensionally identical to each other and to triangle, but I argue that they are intensionally distinct. In this case, overdefinition of the concept triangle produces a third abstract concept that intensionally subsumes its two alternative definitions and has the interesting characteristic

that its necessary conditions are stronger than its sufficient conditions. That is, it is necessary that a triangle have both three sides and three angles, but it is sufficient that it have either three sides or three angles.

## Defining Intensional Subsumption

Since the notion of intensional subsumption allows some latitude for definition, I choose to seek definitions that lead to tractable and efficient algorithms for the common cases that people do effortlessly. In my view, this is the purpose of the representational framework—to efficiently find concepts related to a need. For those cases that require subtle reasoning to infer that any  $x$  is also a  $y$ , I will not expect the classification system to find this out. Rather it must be possible to record this fact and subsequently use it, once it has been discovered by whatever means. The intensional subsumption criterion that I have been exploring is that every element of the parent description subsumes some corresponding element of the child. For example:

[a person with a professional spouse]  
subsumes  
[a woman with a doctor husband]  
because [person] subsumes [woman] and  
[professional spouse] subsumes [doctor husband].

For natural kind concepts and abstract concepts, comparing the sufficient conditions of the parent with the necessary conditions of the child, easily generalizes this subsumption algorithm to handle gap predicates in partial definitions and also the overdefined abstract concepts like triangle.

## Implicit Quantification

Having introduced the idea of intensional subsumption, let me now turn briefly to another topic that interacts with the subsumption algorithm. I have noticed when looking at semantic network systems that many links have implicit quantificational import that is not overtly indicated but is hidden in the meaning of the link and in many cases never explicitly spelled out. For example:

“Birds have wings” means that every bird has some wings.

“People need vitamins” means that every person needs every vitamin.

“People live in houses” may mean that every person has a house to live in, or that houses are things that people live in, or that people are the kinds of things that live in houses, or that

living is what people do in houses (and perhaps more), although it isn’t clear which.

When this kind of quantification is hidden inside the meaning of a link, it is impossible for a classification algorithm to handle links correctly without accessing the full definition of the link and working it out, or else having specific information about specific link names (either wired in or stored somewhere). Moreover, if this information is associated with the link by means of the link name, then sometimes one needs to coin different link names for the same underlying domain relationship in order to have links with different quantificational import. This problem exists not only for link-based network representations, but also for fields in records of databases, for slots in frame systems, for class and instance variables in object-oriented systems, and for many axiomatizations of knowledge in predicate calculus notations.

## Semantic Tags can Reveal Quantificational Import

To solve this problem, in my “Understanding Subsumption” paper (Woods 1991), I introduced the notion of an explicit tag that can make this kind of quantification overt and thus make it visible to the classification algorithm without having to unpack the domain-specific semantics of the rest of the link. For example:

Birds have wings  
is Bird (AE have wing)  
People need vitamins  
is Person (AA need vitamin)  
Children like candy  
is Child (TAA like candy)  
John lives in Chicago  
is John (II live\_in Chicago)

where AE (for all-exists) is the tag that says every instance of the subject has the specified relation to some instance of the object, AA (for All-All) asserts that every instance of the subject has the specified relation to every instance of the object, TAA (for Typically-All-All) means that typically every instance of the subject has the specified relation to every instance of the object (but that exceptions are possible), and II (for instance-instance) specifies that the specified relation holds directly between the subject and object as instances. (These are only illustrative examples of the kinds of things that can be made explicit by such tags.)

These quantificational tags can be treated as relation-forming operators. For example, the tags used earlier could be defined as shown in figure 4.

## Advantages of Explicit Semantic Tags

The advantage of an explicit semantic tag is that it separates the logical quantification information (which the classifier needs to know) from the domain-specific elements of a link (which the classifier can treat as a “black box”) and thus provides the classifier with the needed information to make subsumption decisions without having to know any domain-specific information. It also forms a contract between a knowledge engineer and the reasoner, so that a knowledge engineer can record a fact using a domain-specific relation that is free of excess quantificational baggage and can explicitly indicate the quantificational import that the link is intended to have, without the knowledge engineer or the system having to somehow know and remember which links have what quantificational consequences. I have seen cases in large-scale knowledge representations in which the same underlying domain-specific relation is given two slightly different names, each with slightly different quantificational import, and there is no system to flag that this is going on or to help the knowledge engineer know or remember which relation to use for which purpose. An explicit quantificational tag solves this problem.

Since these tags can be treated as relation-forming operators in link-oriented structures, it should be possible to retrofit this perspective onto existing systems that allow definitions of links, slots, fields, and so on. Figure 5 illustrates some examples of assertional link tags that could be added to links to make their intended quantification clear.

## Subject Restriction and Object Restriction

Similar tags can be used to express assertions about subject restrictions and object restrictions of relations:

VR [person] [live in] [place]

“If a person lives in  $y$  then  $y$  is a place”

SR [person] [live in] [apartment]

“If  $x$  lives in an apartment then  $x$  is a person”

(The latter is not true, of course, but that’s what the SR link means.)

## Tagging the Kind/Instance Distinction

Semantic link tags can also indicate whether they are treating the concept at the end of a

AE(REL) =  
(lambda (X Y) (for every x in X thereis  
(for some y in Y thereis x REL y)))

AA(REL) =  
(lambda (X Y) (for every x in X thereis  
(for every y in Y thereis x REL y)))

TAA(REL) =  
(lambda (X Y) (for every x in X thereis  
(for every y in Y thereis  
(unless known(not(x REL y)) thereis  
likely(x REL y))))))

Figure 4. Quantificational Tags can be Treated as Relation-Forming Operators.

AE [person] [live in] [place]  
“Every person lives in some place.”  
AA [person] [need] [vitamin]  
“Every person needs every vitamin”  
EE [person] [break] [window]  
“Some person breaks some window”  
EA [student] [take] [course]  
“Some student takes every course”

Figure 5. Some Examples of Assertional Link Tags.

link as a kind or as an instance. For example:

AI—“everybody likes John”

EI—“somebody likes John”

IA—“John likes everybody”

IE—“John likes somebody”

SRI—“only sport fans like John”

IVR—“John eats only vegetarian food.”

II—“John likes Mary”

## Structural Link Tags

My “Understanding Subsumption” paper also introduces structural link tags (starting with M, for modifier) that allow one to build up descriptions of structured concepts from constituent elements using links. Examples are:

IKO [student] / (ME [take course] : [math course])

“a student who takes some math course”

```

AE [person] [live in] [place]
  "Every person lives in some place."
AA [person] [need] [vitamin]
  "Every person needs every vitamin"
EE [person] [break] [window]
  "Some person breaks some window"
EA [student] [take] [course]
  "Some student takes every course"

```

Figure 6. The MR Tag Has Inverse Relational Subsumption.

```

IKO [student] / (MA [take course] : [math course])
  "a student who takes every math course"
IKO [student] / (MR [take course] : [math course])
  "a student all of whose courses are math courses"
IKO [student] / (MI [has major] : [mathematics])
  "a student whose major is mathematics"

```

Here, the ME, MA, and MR relationships say that the link is part of the meaning of the concept to which it is attached, as opposed to something that is asserted about it. ME (for "Modifier Exists") says that there exists a concept of the indicated type which has the indicated relationship to the concept being defined, while MA (for "Modifier All") says that every concept of the indicated type has the indicated relationship to the concept being defined. MR (for "Modifier Restriction") says that any object related to the concept being defined by the indicated relationship must be of the indicated type. MI (for "Modifier Instance") says that the specified object concept itself is related to the concept being defined by the specified relationship.

### Semantic Tags Can Answer "What's in a Link?"

With the machinery of semantic tags introduced earlier, we can now have a principled methodology for systematically answering the questions raised in "What's in a Link?" For example, in the black telephone case, the node:

```

N12368
  MI kind-of telephone
  MI color          black

```

would represent the general concept of a black telephone, while:

```

N12368
  II name          telephone
  AI color         black

```

would represent a concept whose name is tele-

phone (that is, the concept of a telephone, if there is only one, or a sense of the word, if there are more than one) with the recorded assertion that all telephones (or at least all N12368s) are black.

## Subsumption Depends on the Meanings of Link Tags

The reason to introduce all this detail about link tags is that subsumption depends on the meanings of the link tags as well as the subsumption relations between the domain-specific concepts involved. For example:

```

[person] / (MR [has son] : [professional])
  subsumes
[woman] / (MR [has child] : [doctor])

```

because of the semantics of the MR quantificational tag and the subsumption relationships between the domain-specific concepts. The meaning of the MR tag has the unexpected consequence, illustrated in figure 6, that the required subsumption relationship between the relations [has son] and [has child] goes in the opposite direction from the usual subsumption dependencies between the elements of a subsuming concept and those of its subsumees. If the semantic tags were both ME, then this relationship would go in the usual direction. That is, a woman who has a son who is a doctor (the ME case) therefore has a child who is a doctor, but a woman all of whose sons are doctors (the MR case) might still have a child (that is, a daughter) who is not a doctor.

This counterintuitive behavior doesn't arise in simpler subsumption systems in which subsumption between elements of descriptions is allowed only when the relations in the two elements are the same. Moreover, most of the other semantic tags have the expected directionality of subsumption between the subsuming and subsumed elements, so this issue is rarely noticed or appreciated. You can imagine why some early expert systems designers were mystified by occasional situations in which inheritance seemed to work backwards.

## Practical Subsumption Algorithms

Now that we have an intensional subsumption criterion that is able to handle partially defined and overdefined abstract intensional concepts, and that appears to be more computationally tractable than extensional subsumption, the next question is whether we can in fact use these structures and notations to support practical and scalable subsumption algorithms

capable of dealing with large knowledge bases. The important issue is not whether the testing of an individual subsumption relationship between two concepts is efficient, but whether the entire process of finding the place in a conceptual taxonomy where a new concept belongs is efficient—that is, can we get efficient MSS and MGS algorithms. I gave arguments in the “Understanding Subsumption” paper why I believe this is possible. A key observation is that the taxonomy is narrow at the top and fans out downward, so that paths searching upward in the structure are generally bounded, while searching downward can become combinatoric. In the paper, I argued that if the downward branching ratio can be controlled, then an appropriate MSS algorithm should be able to look up the most specific subsumers in sublinear time, and that although the MGS algorithm is less well behaved, its average expected time should be sublinear as well. This should make it possible to handle very large taxonomies. It remained to be determined empirically if this is so.

At this point, I wanted to identify a natural population of concepts that could number into the millions with which to test these ideas empirically. I didn’t expect to learn what I wanted from synthetically constructed concept definitions such as randomly generated conceptual structures—I wanted to know what a real natural population of concepts was like. Eventually I settled on using concepts defined by words and phrases extracted from unrestricted natural English text as a natural population of concepts with which to test out these ideas. It turned out that for a wide variety of text sources, the downward branching of the induced taxonomy of natural language words and phrases was naturally bounded, and the MSS and MGS algorithms were empirically practical. Moreover, the resulting conceptual taxonomies turned out to be useful structures for improving text retrieval and browsing. This was discovered in the course of a project that I began at Sun Microsystems Laboratories that attempted to determine whether this kind of conceptual taxonomy could improve information retrieval (Woods 1997, Woods 2000a, Woods 2004).

## Conceptual Indexing

The methodology that I began exploring at Sun is something that I called conceptual indexing. In addition to the usual inverted files that record where individual words and phrases occur in documents, conceptual indexing produces an induced taxonomy of structured con-

cepts organized by intensional subsumption for all of the words and phrases found in the material. This taxonomy also includes additional subsuming concepts that can be inferred from phrase structure and from dictionary entries. The conceptual taxonomy is then used to support searching and browsing. The query engine can use this taxonomy to connect what you ask for to what you need to find. The combination of the inverted files of word and phrase occurrences with the resulting derived conceptual taxonomy is referred to as a *conceptual index* of the material. Coupled with a dynamic passage retrieval algorithm that locates specific passages where the information you need is likely to be found, this methodology has proven to be extremely effective in promoting human search productivity (Ambroziak and Woods 1998, Woods et al. 2000b). The conceptual taxonomies also proved to be an intuitive structure for browsing and navigating. In a separate validation of the use of such taxonomies, a project at Boeing, using a large manually generated taxonomy with automatic augmentation and similar subsumption technology, demonstrated improvement in people’s ability to locate experts in highly technical subject areas (Clark et al. 2000).

The conceptual indexing project at Sun Microsystems began with the question of whether a conceptual taxonomy of the words and phrases from indexed text could be used to improve information retrieval. The project was aimed at improving online information access. The object was to find specific passages of text in response to specific requests, in order to help people find specific information in a timely fashion. This was a very different goal from traditional document retrieval (and it predated and presaged the advent of Web search engines). I argued that if we could significantly improve this kind of online access, it would have wide applicability, and this is turning out to be quite true. (Also, the project would allow me to gain experience with large-scale natural populations of structured concepts.) In the course of this project, systems that we built have been used to construct structured conceptual taxonomies in a variety of subject domains, the largest of which generated over 3 million concepts.

The basic idea of conceptual indexing is to allow people to search by generality, specifying the level of generality of interest by the query terms they choose and relying on the system to use information from the constructed conceptual taxonomy to make connections between terms in the request and related (more specific)

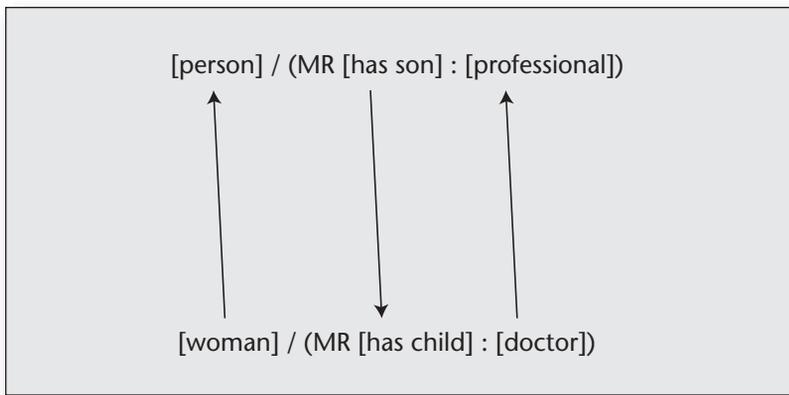


Figure 6. The MR Tag Has Inverse Relational Subsumption.

terms that should be considered hits when found in the text. The conceptual taxonomy can integrate various kinds of subsumption relationships such as syntactic, semantic, and morphological relationships as well as other relationships, such as entailment and containment relationships. For example, in a conceptual index of a bug-description database, the query “color change,” which was not in the taxonomy, was found to subsume the indexed phrases “becomes black,” “reset bitmap colors,” and “color disruption.” Figure 7 gives an illustration of how this last subsumption is derived. On the left side of the figure, “color disruption” is under “disruption” because “disruption” is the head of the phrase. The concept “disruption” is under “change” because it was analyzed morphologically as being derived from the root “disrupt,” and the system’s dictionary knows that disrupt is a kind of damage and that damage is a kind of change. On the right side, “color change” subsumes “color disruption” because “color” subsumes “color” and “change” subsumes “disruption” by virtue of the path on the left.

The morphological relationships that are incorporated into the taxonomy, as well as semantic and other relationships, are drawn from a lexicon of known words with information about their meanings, derivations, and relationships to other words. This lexicon is supplemented during indexing by a morphological analysis component whose job is to construct a lexical entry for any unknown word that the indexer encounters (Woods 2000b). This new lexical entry, which contains syntactic part-of-speech information and may contain morphological information and semantic and other relationships, is then available for immediate use and for any subsequent occurrences of the same word. Information

about the word can then be checked to see whether it should be used to start a potential phrase or whether it can extend a potential phrase already under consideration. As each word and phrase is encountered in the indexed material, if it is not already in the conceptual taxonomy, it is added to the taxonomy and various related terms derived from it or drawn from its lexical entry are also added to the taxonomy with appropriate relationships to the indexed term. The result is a custom taxonomy, every concept of which either occurs in the indexed material or is induced by terms in the indexed material.

A feature of the intensional subsumption perspective is the ability to integrate a variety of abstract subsumption relationships, using link labels to distinguish different kinds of subsumption. Inheritance of properties can then be governed by the sequence of labels in a subsumption path. The conceptual indexing system at Sun used several such subsumption relationships, the most general of which is an entailment relationship in which a term in the indexed text entails the implicit presence of another concept in the context being described. For example, the term *display* entails the concepts of “see” and “visible” although *display* is not a kind of “seeing” or “visibility.” The idea is that if a concept of displaying is present in a context, then the concepts of seeing and visibility are intensionally implicit somewhere in the context. Thus, if someone searches for “see diagram,” a hit for “display diagram” could be found. Similarly, terms derived morphologically from another term entail the implicit intensional presence of the underlying root term, even when the derivation involves significant qualification or even negation. For example, the term *indestructible* entails the implicit presence of the concept of destruction. Specializations of this kind of entailment include the use of morphological relations such as “derived-from” and “inflection-of” as kinds of subsumption relationships, as well as the use of explicit entailment axioms for facts like “plumage” entails “bird.” In a similar way spatial and temporal containment can be treated as kinds of subsumption. Figure 8 illustrates some of the kinds of relationships that can be treated as subsumption and some of the subsumption relationships that hold among them.

Figure 9 illustrates how helpful it can be to browse with a conceptual index. In this case, when searching an index of encyclopedia articles about animals with the query “brown fur,” I found a few subsumed phrases such as “gray brown fur” and “white-spotted brown fur,” but

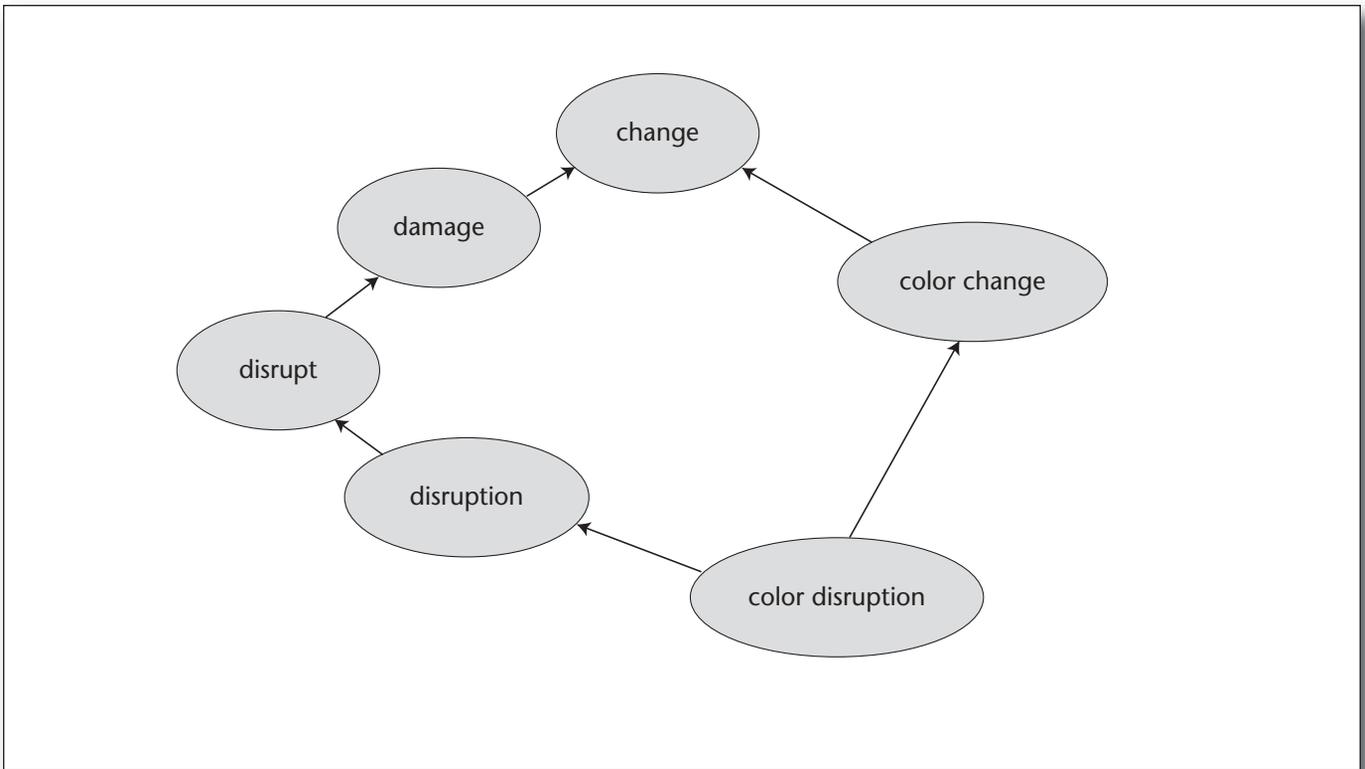


Figure 7. Taxonomy Fragment Illustrating Syntactic, Semantic, and Morphological Relationships.

the system helpfully indicated that “brown fur” was classified in the taxonomy under the concept “brown coat.” Clicking “brown coat” to generalize the query found the concepts shown.

### Discussion and Future Directions

The conceptual indexing application I have just described makes some interesting use of subsumption technology, but it uses it in a very special and limited way that doesn’t come close to the kind of reasoning that I’d like to be able to support. A next step is to move beyond systems for passage retrieval (or similar systems that extract bits of text from source material) toward systems that can do reasoning in support of a user’s information need. This includes general-purpose question-answering systems that can perform useful reasoning and perception, based on the information in the indexed material. General-purpose question answering requires at least an ability to understand passages of text that contain answers or that contain information necessary to infer answers and an ability to efficiently reason with this information. This requires robust natural language parsing, semantic interpretation, finding and using relevant background knowledge,

- ENTAILS
- ISA
- INSTANCE-OF
- KIND-OF
- HAS-ROOT
- INFLECTION-OF
- DERIVED-FROM
- SPECIALIZES
- VARIANT-OF
- NICKNAME-OF
- MISSPELLING-OF
- CONTAINED-IN
- SPATIAL-SUBREGION-OF
- TEMPORAL-SUBINTERVAL-OF

Figure 8. Some Kinds of Subsumption Relationships.

and reasoning with algorithms that can deal with large amounts of knowledge. I believe that the kinds of intensional subsumption techniques that I have described will be helpful in providing an infrastructure for such algo-

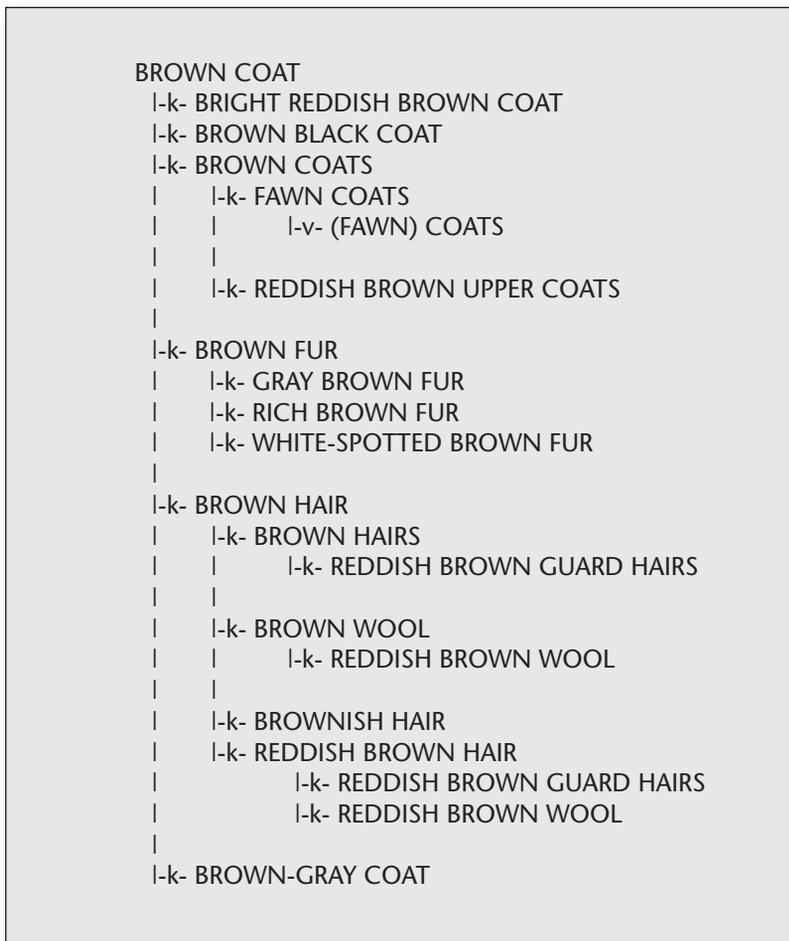


Figure 9. Browsing in a Conceptual Taxonomy.

rithms, not only in the organization of conceptual taxonomies and the retrieval of text passages, but also in the organization of facts and rules necessary to support inference with large-scale knowledge bases. This will require continued development of the theory and practice of scalable knowledge representation technology, such as understanding the circumstances under which downward branching in a conceptual taxonomy will be bounded or the algorithms will be otherwise well behaved and how subsumption technology can be integrated with probabilistic and rule-based inference.

With the increasing availability and use of online information, the inability of searching alone to meet many of people's information needs is becoming more and more apparent. Accordingly, question answering has once again become a major research topic, especially in the areas of open-domain question

answering where the resource against which questions are answered consists of large text collections with virtually unrestricted content. Most work in this area involves transforming a question into a search query, followed by a phase of paragraph or passage retrieval (usually preceded by a document retrieval phase to find documents to search for passages), followed by a phase of answer extraction from the identified passages. Conceptual indexing appears to be a good candidate for the passage retrieval phase of this process, since it avoids the need for an intermediate document retrieval phase, and it directly handles many of the paraphrase issues. Preliminary attempts to explore this hypothesis by participating in the TREC Question Answering Track (Woods et al. 2000a, Woods et al. 2001) have been only partially successful due to technical issues and limitations of the query formulation and answer extraction components that were used, but they did manage to establish that this kind of subsumption knowledge can help and that more of it helps more. This is consistent with some other QA track results (Harabagiu et al. 2001) where similar lexical and semantic information derived from WordNet was used with a more sophisticated answer-extraction component. These explorations also revealed that conceptual indexing is a powerful tool for investigating the issues that arise with this task. For example, being able to quickly probe the corpus and discover that an answer to a question about a Russian astronaut is answered in a passage about a Soviet cosmonaut reveals some of the paraphrase issues that are involved.

Perhaps the most interesting thing about the TREC QA-track questions is the degree to which background knowledge is required to answer many of them and the diversity of the background knowledge that is required. For example the question "What are the names of the tourist attractions in Reims?" finds no passages with "Reims" that involve tourists or attractions or seeing or visiting. One has to know that a cathedral is something that someone might like to visit to recognize that Reims Cathedral is an answer. On the other hand, finding the answer passage for "What is the longest word in the English language?" is easy, given the way the answer passage is worded, but answering the paraphrase "What English word contains the most letters?" would require knowledge of how word lengths are calculated and the ability to do some sophisticated reasoning. While it is clearly a logistical problem to acquire the necessary background knowledge for this task, a challenging prerequisite is figuring out a way to organize and use that

knowledge that will scale to the size of the task. It is in this area that I think generalizations of the subsumption technology described here and other link-based algorithms have the most promise.

My “What’s in a Link” paper has sometimes been taken as an argument against semantic networks. It should now be clear that my goal is quite the opposite, namely to make semantic networks sufficiently rigorous to meet the challenges that I raised, while keeping the advantages of associative link-based representations for efficient reasoning (at least for those things that people do efficiently). One system that has attempted to combine rigor with link-based representations is the system of Stuart Shapiro (Maida and Shapiro 1982). Another is James Crawford’s system of Access Limited Logic (Crawford 1990). Neither of these systems, however, fully address my needs for a sublinear algorithm for finding the most specific subsumers of a goal description from a large knowledge base. My hope is that the work described here will lead to scalable solutions to these problems in the future.

## Conclusion

These are times of exciting opportunities. The goal of question answering in open-ended domains raises many new problems of representation and reasoning, and many of the old problems have come back and are still with us. For example, a recent search on a collection of *New York Times* articles found a new variant on the classic “What do worms eat?”:

Query: What does BMW make?

Answer: The BMW makes your upward mobility all too obvious.

I have presented a range of issues that relate to what nodes and links in associative networks might mean and how such structures can aid in understanding language and may be used to support intelligent reasoning. I have also presented a number of techniques and methods for dealing with some of those issues in a way that can be used to help people find information in large bodies of knowledge. The resolution of these issues is far from complete, and this article represents only a status report for an ongoing effort. I believe we still need to forge some new tools to solve some of these problems (Woods 1987c). I hope that the ideas presented here will help point the way to new and better solutions. In terms adapted from a saying of Warren McCulloch:

Don’t just look at the end of my finger—look where I’m pointing.

## Acknowledgments

For their inspiration and intellectual stimulation, I would like to thank my many colleagues and students, many of whom have participated in some of these projects. There are far too many of these to list here. I would also like to express my appreciation for the stimulating work environments provided by Harvard, BBN, Apex, ON Technology, Sun Microsystems, and ITA Software. In addition, I would like to thank IBM for sponsoring the series, Great Moments in Knowledge Representation, and Chris Welty for organizing the series and inviting me as one of its speakers. Finally, I would like to acknowledge and thank NSF, NASA, DARPA, and the Kapor Family Foundation for the support of portions of the work described here.

## References

- Ambroziak, J., and Woods, W. A. 1998. Natural Language Technology in Precision Content Retrieval. In *Proceedings of the International Conference on Natural Language Processing and Industrial Applications (NLP+IA 98)*. Moncton, New Brunswick, Canada: University of Moncton.
- Bobrow, D. G. 1964. A Question-Answering System for High School Algebra Word Problems. In *Proceedings of AFIPS Fall Joint Computer Conference*, Vol. 26, 591–614. Montvale, NJ: AFIPS.
- Brachman, R. J. 1977. A Structural Paradigm for Representing Knowledge, Ph.D. dissertation, Division of Engineering and Applied Physics, Harvard University, Cambridge, MA.
- Brachman, R. J., and Levesque, H. J. 2004. *Knowledge Representation and Reasoning*. San Francisco: Morgan Kaufmann Publishers.
- Brachman, R. J., and Levesque, H. J. 1987. Expressiveness and Tractability in Knowledge Representation and Reasoning. *Computational Intelligence* 3(2): 78–93.
- Brachman, R. J., and Schmolze, J. G. 1985. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science* 9(2): 171–216.
- Carbonell, J. R. 1970. AI in CAI: An Artificial-Intelligence Approach to Computer-Assisted Instruction. *IEEE Transactions on Man-Machine Systems*, MMS-11(4): 190–202.
- Clark, P.; Thompson, J.; Holmback, H.; and Duncan, E. 2000. Exploiting a Thesaurus-Based Semantic Net for Knowledge-Based Search. In *Proceedings of the 12th Conference on Innovative Applications of Artificial Intelligence*, 988–995. Menlo Park: AAAI Press.
- Collins, A. M., and Quillian, M. R. 1969. Retrieval Time from Semantic Memory. *Journal of Verbal Learning and Verbal Behavior* 8: 240–248.
- Collins, A.; Warnock, E. H.; Aiello, N.; and Miller, M. L. 1975. Reasoning from Incomplete Knowledge. In *Representation and Understanding: Studies in Cognitive Science*, ed. D. Bobrow and A. Collins, 383–415. New York: Academic Press.
- Crawford, J. M. 1990. Access-Limited Logic—A Lan-



guage for Knowledge Representation. Doctoral dissertation, Department of Computer Sciences, The University of Texas at Austin. (Published as Technical Report AI90-141, Artificial Intelligence Laboratory, The University of Texas at Austin.)

Davis, R.; Shrobe, H.; and Szolovits, P. 1993. What Is a Knowledge Representation? *AI Magazine* 14(1): 17–33.

Harabagiu, S. M.; Moldovan, D. I.; Pasca, M.; Mihalcea, R.; Surdeanu, M.; Bunescu, R. C.; Girju, R.; Rus, V.; and Morarescu, P. 2001. The Role of Lexico-Semantic Feedback in Open-Domain Textual Question-Answering. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, 274–281. Morristown, NJ: Association for Computational Linguistics.

Maida, A. S., and Shapiro, S. C. 1982. Intensional Concepts in Propositional Semantic Networks. *Cognitive Science* 6(4): 291–330.

Quillian, M. R. 1969. The Teachable Language Comprehender: A Simulation Program and Theory of Language, *Communications of the ACM* 12(8): 459–476.

Simmons, R. F. 1965. Answering English Questions by Computer: A Survey. *Communications of the ACM* 8(1): 53–70.

Woods, W. A. 2004. Searching vs. Finding. *ACM Queue* 2(2): 27–35.

Woods, W. A. 2000a. Conceptual Indexing: Practical Large-Scale AI for Efficient Information Access. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000)*, 1180–1185. Menlo Park: AAAI Press.

Woods, W. A. 2000b. Aggressive Morphology for Robust Lexical Coverage. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-2000)*, 218–223. San Francisco: Morgan Kaufmann.

Woods, W. A. 1997. Conceptual Indexing: A Better Way to Organize Knowledge. Tech-

nical Report SMLI TR-97-61, Sun Microsystems Laboratories, Mountain View, CA.

Woods, W. A. 1991. Understanding Subsumption and Taxonomy: A Framework for Progress. In *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, ed. J. Sowa, 45–94. San Mateo, CA: Morgan Kaufmann.

Woods, W. A. 1987a. Grammar, Augmented Transition Network. In *Encyclopedia of Artificial Intelligence*, ed. S. C. Shapiro, 323–333. New York: John Wiley and Sons.

Woods, W. A. 1987b. Semantics, Procedural. In *Encyclopedia of Artificial Intelligence*, ed. S. C. Shapiro, 1029–1031. New York: John Wiley and Sons.

Woods, W. A. 1987c. Don't Blame the Tool. *Computational Intelligence* 3(3): 228–237.

Woods, W. A. 1986a. Important Issues in Knowledge Representation. *Proceedings of the IEEE*, 74(10): 1322–1334.

Woods, W. A. 1986b. Problems in Procedural Semantics. In *Meaning and Cognitive Structure: Issues in the Computational Theory of Mind*, ed. Z. W. Pylyshyn and W. Demopoulos. Norwood, NJ: Ablex Publishing Corporation.

Woods, W. A. 1981. Procedural Semantics as a Theory of Meaning. In *Elements of Discourse Understanding*, ed. A. Joshi, B. L. Webber, and I. Sag, 300–334. Cambridge, UK: Cambridge University Press.

Woods, W. A. 1979. *Semantics for a Question-Answering System*. New York: Garland Publishing.

Woods, W. A. 1978. Semantics and Quantification in Natural Language Question Answering. In *Advances in Computers*, Vol. 17, ed. M. Yovits. New York: Academic Press.

Woods, W. A. 1975. What's in a Link: Foundations for Semantic Networks. In *Representation and Understanding: Studies in Cognitive Science*, ed. D. Bobrow and A. Collins, 35–82. New York: Academic Press.

Woods, W. A. 1970. Transition Network Grammars for Natural Language Analysis. *Communications of the ACM* 13(10): 591–606.

Woods, W. A. 1968. Procedural Semantics for a Question-Answering Machine. In *Proceedings of AFIPS Fall Joint Computer Conference*, Vol. 33, 457–471. Montvale, NJ: AFIPS.

Woods, W. A. 1967. Semantics for a Question-Answering System. Ph.D. dissertation, Division of Engineering and Applied Physics, Harvard University, Cambridge, MA.

Woods, W. A., and Schmolze, J. 1992. The KL-ONE Family. *Computers and Mathematics with Applications* 23(2–5): 133–177.

Woods, W. A.; Bates, M.; Brown, G.; Bruce, B.; Cook, C.; Klovstad, J.; Makhoul, J.;

Nash-Webber, B.; Schwartz, R.; Wolf, J., and Zue, V. 1976. Speech Understanding Systems: Final Technical Progress Report, Volumes I–V, BBN Technical Report 3848, Bolt Beranek and Newman Inc., Cambridge, MA.

Woods, W. A.; Bookman, L. A.; Houston, A.; Kuhns, R. J.; Martin, P.; and Green, S. 2000b. Linguistic Knowledge can Improve Information Retrieval. In *Proceedings of the 6th Conference on Applied Natural Language Processing Conference (ANLP-2000)*, 262–267. San Francisco: Morgan Kaufmann.

Woods, W. A.; Green, S.; Martin, P.; and Houston, A. 2001. Aggressive Morphology and Lexical Relations for Query Expansion. In *Proceedings of the Tenth Text Retrieval Conference (TREC-10)*, 479–485. Gaithersburg, Maryland: National Institute of Standards and Technology.

Woods, W. A.; Green, S.; Martin, P.; and Houston, A. 2000a. *Halfway to Question Answering*. In *Proceedings of the Ninth Text Retrieval Conference (TREC-9)*, 489–500. Gaithersburg, Maryland: National Institute of Standards and Technology.

Woods, W. A.; Kaplan, R. M.; and Nash-Webber, B. L. 1972. The Lunar Sciences Natural Language Information System: Final Report. BBN Report No. 2378, Bolt Beranek and Newman Inc., Cambridge, MA. (Available from NTIS as N72-28984.)



**William A. Woods** (wwoods@itasoftware.com) is internationally known for his research in natural language processing, continuous speech understanding, knowledge representation, and knowledge-

based search technology. He earned his doctorate at Harvard University, where he then served as an assistant professor and later as a Gordon McKay Professor of the Practice of Computer Science. He was the principal investigator for BBN's early work in natural language processing and knowledge representation and for its first project in continuous speech understanding. Subsequently, he was chief scientist for Applied Expert Systems and principal technologist for ON Technology, before joining Sun Microsystems as a principal scientist and distinguished engineer in 1991. He is currently Distinguished Software Engineer at ITA Software in Cambridge, Massachusetts. He is a past president of the Association for Computational Linguistics, a fellow of the Association for the Advancement of Artificial Intelligence, and a fellow of the American Association for the Advancement of Science.