Knowledge Representation

Article reference code 58
Draft of an article to appear in
Encyclopedia of Cognitive Science,
Macmillan Publishers Ltd.

Stuart C. Shapiro
Professor, Department of Computer Science and Engineering
and Member, Center for Cognitive Science
University at Buffalo, The State University of New York
Buffalo, NY 14260-2000, USA
Phone: 716-645-3180 Ext. 125
Fax: 716-645-3464
Email: `shapiro@cse.buffalo.edu`

April 9, 2002

Headword Knowledge representation

Keywords knowledge representation, commonsense knowledge, logic, propositional logic, predicate logic, predicate calculus, procedural representation, production systems, semantic networks, schemas, frames, scripts, analogical representation, connectionism

Contents list

Article definition Knowledge representation is a subarea of Artificial Intelligence concerned with understanding, designing, and implementing ways of representing information in computers so that programs can use this information to derive information that is implied by it, to converse with people in natural languages, to plan future activities, and to solve problems in areas that normally require human expertise.

1 Introduction

Knowledge representation (*see* KNOWLEDGE REPRESENTATION, PSYCHOLOGY OF; REPRESENTATION, PHILOSOPHICAL ISSUES ABOUT) is a subarea of Artificial Intelligence (*see* ARTIFICIAL INTELLIGENCE, PHILOSOPHY OF) concerned with understanding, designing, and implementing ways of representing information in computers so that programs can use this information

- to derive information that is implied by it,
- to converse with people in natural languages,
- to plan future activities,
- to solve problems in areas that normally require human expertise.

Deriving information that is implied by the information already present is a form of *reasoning* (*see* DEDUCTIVE REASONING; LANGUAGE OF THOUGHT; REASONING; REASONING, PHILOSOPHICAL ISSUES ABOUT; REPRESENTATION, IMPLICIT AND EXPLICIT). Because knowledge representation schemes are useless without the ability to reason with them, the field is usually known as "knowledge representation and reasoning."

Many philosophers consider knowledge to be justified true belief. Thus, if John believes that the world is flat, we would not say that John *knows* that the world is flat, because he is wrong—"the world is flat" is not true. Also, it may be that Sally believes that the first player in chess can always win, Betty believes that the second player can always win, and Mary believes that, with optimal play on both sides, chess will always end in a tie. One of them is correct, but we would still not say that any of them *knows* the answer, because their belief cannot have been justified by a complete analysis of the game. A computer system could not limit its information to knowledge in this strict sense, so it would be more accurate to say that the topic being discussed is *belief representation,* rather than knowledge representation. (*See* EPISTEMOLOGY.) Nevertheless, we will continue to use "knowledge representation," because that has become accepted as the name of this subject.

2 Representing commonsense knowledge

The field of knowledge representation began, around 1958, with an investigation of how a computer might be able to represent and use the kind of commonsense knowledge we have when we decide that to get from our house to the airport,

Table 1: Some sentences and how we understand them.

Sentence	**How we understand it**
John likes ice cream.	John likes to eat ice cream.
Mary likes Asimov.	Mary likes to read books by Isaac Asimov.
Bill flicked the switch. *The room was flooded with light.*	Bill moved the switch to the "on" position, which caused a light to come on, which lit up the room Bill was in.
Betty opened the blinds. *The courtyard was flooded with light.*	Betty adjusted the blinds so that she could see through the window they were in front of, after which she could see that the courtyard on the other side of the window was bright.

we should walk to our car and drive to the airport, rather than, for example drive to our car and then walk to the airport.

In the 1960s and 1970s, much knowledge representation research was concerned with representing and using the kind of information we get from reading and from talking to other people; that is, the information that is often expressed in natural languages, and that underlies our ability to understand and to use natural languages. (*See* MEANING; NATURAL LANGUAGE PROCESSING; SEMANTIC MEMORY: COMPUTATIONAL MODELS.) For example, we probably understand each of the sentences in the first column of Table 1 as shown in the second column, by adding our "background knowledge" to what the sentences explicitly say. Moreover, our understanding of English includes our being able to make the following inferences.

> *Every student studies hard. Therefore every smart student studies.*
>
> *Tuesday evening, Jack either went to the movies, played bridge, or studied. Tuesday evening, Jack played bridge. Therefore, Jack neither went to the movies nor studied Tuesday evening.*

In the 1970s and 1980s, researchers became increasingly concerned with knowledge about specific domains in which human experts operate, such as medical diagnosis and the identification of chemical compounds from mass spectrometry data (*see* EXPERT SYSTEMS; EXPERTISE), and also with the other extreme—knowledge about the everyday world that everyone knows, such as the fact that when you tip over a glass of water, the water will spill on the floor.

In the 1980s and 1990s, these concerns focussed on the details of specific subdomains of everyday knowledge, such as theories of time and space (*see* SPATIAL REPRESENTATION AND REASONING), and also on the general structure of our knowledge of everyday terms, leading to the construction of large and general purpose "ontologies." For example, the Cyc Project has devoted many staff-years to the organization of a computer-useable representation of all the knowledge that is *not* contained in encyclopedias (Thus the name "Cyc," from "en**cyc**lopedia.") but is assumed to be already known by people who read them, and Lycos is using such an ontology to organize searches of the World-Wide Web.

All these threads continue into the 2000s.

3 Predicate calculus and other logical representation schemes

In the late 1800s and early 1900s, various formal systems were developed by people who hoped, thereby, to turn human reasoning into a kind of calculation (*see* INFERENCE USING FORMAL LOGICS). From our perspective, we can now see that what these people were engaged in was research in knowledge representation. The formal systems they developed were systems of logic, a topic which has been studied since the days of Plato and Aristotle. We may consider logic to be the study of correct reasoning. The systems of logic developed in the late 1800s and early 1900s consist of three basic components:

syntax: the specification of a set of atomic symbols, and the grammatical rules for combining them into well-formed expressions;

semantics: the specification of the meaning of the atomic symbols, and the rules for determining the meanings of the well-formed expressions from the meanings of their parts;

proof theory: the specification of a set of rules, called "rules of inference", which, given an initial collection, called a "proof," of well-formed expressions, called "axioms," specify what other well-formed expressions can be added to the proof.

There are two kinds of "meaning" determined by the semantics of a system of logic. In one, we might say that the meaning of G is the claim made by the English sentence, "The moon is made of green cheese." For this notion of meaning, the meaning of $\neg G$, as shown in Table 2, would be the same as "It is not the case that the moon is made of green cheese," or "The moon is not made of green cheese." The other sense of "meaning" is a truth value. Different systems of logic have different truth values, and even different numbers of truth values. There are two-valued logics, three-valued logics, four-valued logics, and even logics with an infinite number of truth values. Two-valued logics usually call their truth values "True" and "False." Some logicians would say that in such a two-valued logic any sentence either means True or means False. Less strictly, one might say that in such a logic, the semantics assigns a truth value of True or False to every sentence. In this notion of meaning, if some sentence P happened to be (or be assigned the truth value of) True, then $\neg P$ would be (or be assigned the truth value of) False, and if P were False, than $\neg P$ would be True. So, if G meant (by the first sense of meaning) "The moon is made

Table 2: A set of propositional connectives and their meaning.

Propositional Connectives	**Sample Use**	**Meaning**
$\neg$	$\neg P$	It is not the case that P.
$\wedge$	$P \wedge Q$	P and Q.
$\vee$	$P \vee Q$	P or Q, or both.
$\Rightarrow$	$P \Rightarrow Q$	If P then Q

Table 3: The quantifiers and their meanings.

Quantifier	**Sample Use**	**Meaning**
$\forall$	$\forall x P(x)$	Every x is a P.
$\exists$	$\exists x P(x)$	Some x is a P.

of green cheese," then, G would be (or mean, in the second sense of meaning) False, so $\neg G$ would be (or mean, or have the truth value of) True.

Although the proof theory considers only the syntax of the expressions, not their semantics, it is usually the case that if the semantics assigns a truth value of True to the axioms, all expressions that the rules of inference add to the proof will also be True. Logics that have this property are called **sound**. Soundness seems to capture the notion of correct reasoning, which is what the study of logic is all about.

Many different logics have been described and investigated. Propositional (or "sentential") logics (*see* REPRESENTATION USING SENTENTIAL LOGICS) do not analyze information below the level of the proposition (or sentence), but use "propositional connectives," such as are shown in Table 2 to build more complex sentences from simpler sentences. For example, the sentence "*Students who study hard get good grades.*" could not be represented in more detail than $P \Rightarrow Q$, where P represents "*Students study hard.*" and Q represents "*Students get good grades.*" First-order logics (predicate logics) continue the analysis down to objects, classes, properties, and relations, with the aid of the quantifiers shown in Table 3. So in some first-order logic, "*Students who study hard get good grades.*" might be represented as $\forall x(Student(x) \wedge study(x, hard) \Rightarrow get(x, grades, good))$. Some first-order logics allow functions. In one of them, this sentence might be represented as $\forall x(Student(x) \wedge study(x, hard) \Rightarrow get(x, good(grades)))$. Second-order logics allow functions, classes, properties, and relations, themselves, to be the arguments of other functions, classes, properties, and relations. In one of them, this sentence might be represented as $\forall x(Student(x) \wedge hard(study)(x) \Rightarrow get(x, good(grades)))$.

A logical sentence that always evaluates to True regardless of the meanings of its atomic parts is called **valid**. In most standard logics, the sentence $P \wedge \neg P \Rightarrow Q$ is valid, meaning that a contradiction implies anything whatsoever, but in some logics, called "paraconsistent" logics, that sentence is not valid. In most standard logics the sentence $P \vee \neg P$ is valid, meaning that any sentence is either

True or False, but in some logics, called "intuitionistic" logics, that sentence is not valid.

Someone who uses a propositional logic to formalize some domain of interest chooses the proposition symbols to be used to represent the sentences of the domain, and their semantics—what sentence each symbol will represent. Someone who uses a predicate logic to formalize some domain of interest chooses the syntax and semantics of the individual constants that represent objects in the domain, the function symbols that represent functions in the domain, and the predicate symbols that represent classes, properties, and relations in the domain. The logic, itself, determines the propositional connectives and the quanitifiers, and how they are to be used, along with function and predicate application, to determine the nonatomic expressions, and their meaning. The rules of inference also operate only on nonatomic expressions, and pay attention only to the logical constants. This is the sense in which people consider these logics to be "formal" logics that pay attention only to the form, and not to the content, of the logical expressions. When one chooses a logic to use, one is choosing the formal apparatus supplied by that logic.

Any knowledge representation and reasoning system consists of two parts—a knowledge representation language and a reasoning component. If the knowledge representation language is well-defined, it will have a well-defined syntax to describe the atomic symbols and how well-defined symbol structures may be constructed from them, and a well-defined semantics to describe what the atomic symbols and the symbol structures are supposed to mean. The reasoning component is a program, often called an "inference engine," that, given a "knowledge base" of symbol structures, adds additional symbol structures to that knowledge base according to the rules implemented in the program. Clearly these components—syntax, semantics, inference engine, knowledge base—correspond to the components of logics—syntax, semantics, proof theory, proof. So we may view any knowledge representation and reasoning system as a logic, and ask about it what kind of logic it is, what formal apparatus it supplies, and whether or not it is sound. The user of a knowledge representation and reasoning system, like the user of a logic, must choose a system, and then design the representations that are not at the level of knowledge representation constructs that the system deals with. In the knowledge representation world, this person is called a "knowledge engineer."

4 Procedural representations

In the mid-1970s, knowledge representation researchers were embroiled in what was called the "declarative/procedural controversy." Although this controversy has largely been resolved (in a sort of compromise), it is worthwhile understanding these two approaches to knowledge representation.

First, we must recognize that there are several kinds of knowing, among which are *knowing that, knowing who,* and *knowing how. Knowing that* is the kind of knowledge we have of propositions. For example, we may *know that*

Seattle is north of San Francisco. *Knowing who* is acquaintance with a person, animal, object, etc. We may say that we know a person, even though we might not know some important facts about that person, for example her birthdate. On the other hand, we may know many facts about a person without being able to say we know that person. For example, many of us know many facts about Bill Clinton, but how many of us can truly say, "I know Bill Clinton"? *Knowing how* is knowledge of how to do things, for example how to swim or ride a bicycle.

There has not been much work in knowledge representation on *knowing who,* and everyone would agree that a procedural representation is appropriate for *knowing how,* though more on this later. The declarative/procedural controversy was about how to represent *knowing that.* The declarativists were in favor of representing propositions that are known (believed) by some agent as a symbol structure with declarative semantics, for example a well-formed expression of propositional or predicate logic, stored in the agent's knowledge base. The proceduralists were in favor of representing such propositions as small programs. For example, when the early (simulated) robot SHRDLU (*see* SHRDLU) was told, "I own blocks which are not red, but I don't own anything which supports a pyramid," it represented that information as two small procedures in the PLANNER programming language. When, later, SHRDLU was asked "do I own anything in the box?", it ran those two procedures to determine the answer. The problem with maintaining this distinction between declarative and procedural representations of knowing-that is the well-known equivalence of data and program. A procedure can be written in a declarative programming language such as Lisp or Prolog, and can thereby be viewed as a symbol structure with declarative semantics. On the other hand, when a declarative representation is used by the inference engine to draw some inference, the declarative representation may be viewed as a program in the programming language interpreted by the inference engine. In this sense, whether a representation is declarative or procedural depends on how one views it.

We might resurrect the declarative/procedural distinction by considering our own knowledge of how to do things. Many of us know how to ride a bicycle. However, few of us can describe how to ride a bycicle, for example in order to instruct someone else. We might consider this knowledge, then, procedural *only*. In this view, all knowledge may be viewed as procedural knowledge, but only knowledge that can be expressed in a declarative language by the knower may be viewed as declarative knowledge. As another example, the restaurant script (see below) is a representation of what typically happens in a restaurant. There have been programs that, supplied with the restaurant script, could fill in details about what happened in restaurant stories. For example, given the story, "John went to a restaurant and ordered a steak," such a program could infer that John was seated and given a menu between entering and ordering. (*See* STORY UNDERSTANDING.) However, most of these programs could not answer questions about the restaurant script itself, such as "What typically happens in a restaurant after the patron is seated?" It is, therefore, reasonable to say that for these programs the restaurant script is not represented declaratively, but only procedurally.

5 Production systems

Production systems (*see* PRODUCTION SYSTEMS AND RULE-BASED INFERENCE; RULE-BASED THOUGHT) are a subclass of knowledge representation and reasoning systems. The knowledge base of a production system is divided into two parts, a working memory (*see* WORKING MEMORY: COMPUTATIONAL MODELS OF) and a rule memory. The working memory consists of a set of symbol structures not containing variables. The rule memory consists of a set of pattern-action rules. Each pattern-action rule has a "left-hand side," which is a set of patterns, and a "right-hand side," which is a set, or sequence, of actions. The patterns and actions may contain variables as long as every variable in the right-hand side of a rule is also in the left-hand side of the same rule. If every pattern in a rule matches some symbol structure in working memory, with a consistent substitution of constants for the variables in the patterns, then the rule is said to be "triggered." A triggered rule may "fire," in which case every action in the right-hand side is performed after replacing the variables with the constants they were matched to in the left-hand side. The actions allowed in the right-hand sides of rules vary among different production systems, but they generally include adding structures to working memory and removing structures from working memory. They may also include adding and removing rules, and interacting with a user.

Since the firing of a rule may change the structures in working memory, it may also change which other rules are triggered. So if several rules are triggered at the same time, the ultimate behavior of the production system may depend on which triggered rule fires first. This is determined by a conflict resolution strategy. Typical strategies are: don't fire any rule more than once on the same variable substitutions; fire the rule that was triggered most (or least) recently; fire a more specific rule before a less specific rule.

Production systems were first designed to be a model of the human mind. For this purpose, the size of working memory and the allowed actions were restricted. However, they have been a popular architecture for expert systems, for which purpose those restrictions were lifted.

It should be noted that the symbol structures in working memory and the patterns and actions in rule memory must be formulated in some knowledge representation language. What the production system architecture provides is a particular style of reasoning and acting using that language.

6 Semantic networks

Semantic networks (*see* SEMANTIC NETWORKS) are a variety of labeled, directed acyclic graph in which the nodes denote entities and labeled directed arcs denote relations between the nodes they connect. Two kinds of semantic networks have

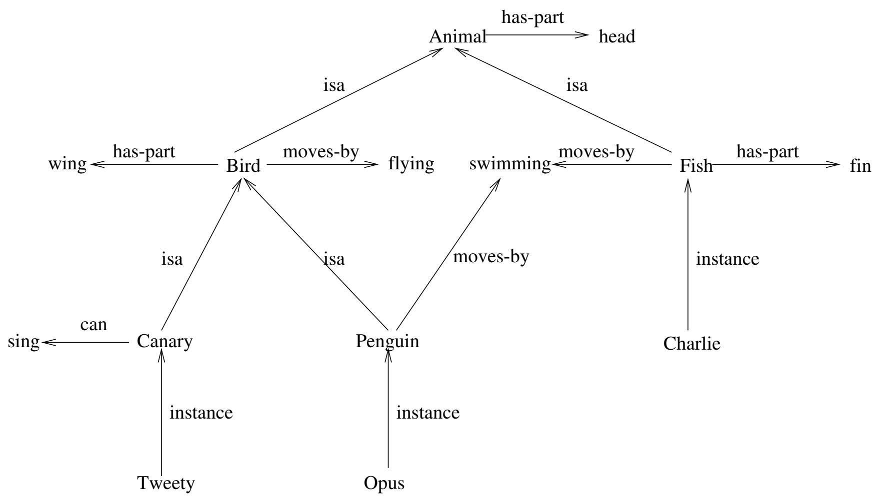

Figure 1: An inheritance-style semantic network

been developed, inheritance networks (*see* INHERITANCE) and propositional semantic networks. Figure 1 illustrates an inheritance network which is intended to represent the following information:

> *Birds and fish are animals. Canaries and penguins are birds.*
> *Animals have heads. Birds have wings. Fish have fins.*
> *Birds move by flying. Fish and penguins move by swimming.*
> *Canaries can sing.*
> *Tweety is a canary. Opus is a penguin. Charlie is a fish.*

Notice that some of the nodes represent categories (`Animal`, `Bird`, `Fish`, `Canary`, and `Penguin`), and others represent individuals (`Tweety`, `Opus`, and `Charlie`). The relation between an individual and the categories it is a member of (`instance`) is different from the relation between a category and its supercategories (`isa`).

Early presenters of inheritance networks did not make the semantics of the relations very clear. For example, it is not clear in Figure 1 what it means when arcs with the same label emanate from a category and one of its supercategories. Surely, birds have both wings and heads, but penguins swim and don't fly. Moreover, although the `has-part` relation seems simple, the `has-part` relation from `Animal` to `head` must mean *Every instance of Animal has a part which is an instance of head.* These semantic confusions were clarified in the successors to simple inheritance networks, the most prominent of which are called Description Logics.

Description Logics are a variety of inheritance networks in which categories (called "concepts" (*see* CONCEPTS, PHILOSOPHICAL ISSUES ABOUT; CONCEPTUAL REPRESENTATIONS IN PSYCHOLOGY.)) and relations (called "roles") between them can be defined without the semantic confusions of earlier inheritance networks. For example, using a formalism called $\mathcal{KL}$, which was designed to reflect many of the features common to different description logics, a parent can be defined as a person with at least one child who is also a person, as follows.

```
(cdef PARENT (and PERSON (c-some Child PERSON)))
```

Here, PARENT is the concept being defined, PERSON is a concept which, presumably, has already been defined, and Child is a role. This is also an example of a concept defined with necessary and sufficient conditions. That is, if Ken is said to be a PARENT, it is *necessary* that Ken be a PERSON with at least one Child who is a PERSON. So the description logic system can infer that Ken is a person, has at least one child, and that that child is a person. On the other hand this same definition says that if Judi is a PERSON with at least one Child who is a PERSON, that is *sufficient* information to conclude that Judi is a Parent. Natural kinds, such as birds, fish, and animals (*see* NATURAL KINDS AND ARTIFACTS), cannot be given necessary and sufficient conditions, so *primitive concepts* can be defined with only necessary conditions. The $\mathcal{KL}$ definitions of ANIMAL and FISH from Figure 1 are

```
(cprim ANIMAL (and top
                   (c-some Part HEAD)
                   (c-atmost 1 Part HEAD)))
(cprim FISH (and ANIMAL
                 (c-some Part FIN)
                 (c-some Moves-by SWIMMING)))
```

This says that every FISH has one head, by inheritance from ANIMAL, and, in addition, has one or more FINs. Since description-logic roles accumulate in this way, the only way to say that birds fly, but penguins are birds that swim instead, is to separate flying birds from swimming birds:

```
(cprim BIRD (and ANIMAL
                 (c-atleast 2 Part WING)
                 (c-atmost 2 Part WING)))
(cdef FLYING-BIRD (and BIRD
                       (c-some Moves-by FLYING)))
(cprim PENGUIN (and BIRD
                    (c-some Moves-by SWIMMING)))
(cprim CANARY (and FLYING-BIRD
                   (c-some Can SING)))
```

All these $\mathcal{KL}$ constructs define concepts, and are considered part of the description logic *terminological component*. To actually make assertions about individuals, most description logics also have an *assertional component*. Assertions in the assertional component are usually written in a syntax that looks like normal first-order predicate logic in which defined concepts can be used as unary predicates and defined relations can be used as binary relations. For example, we might have

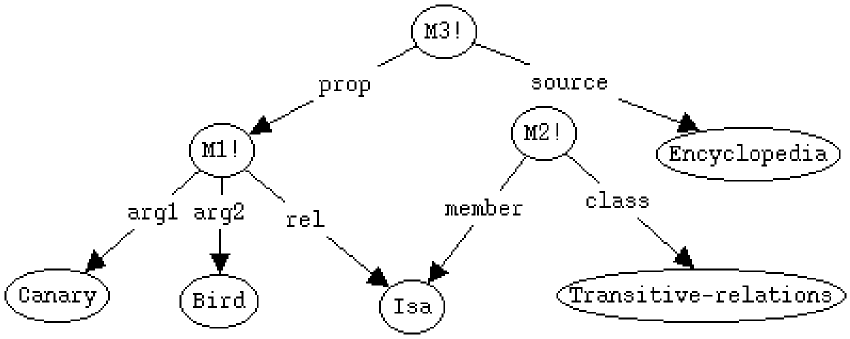

Figure 2: A propositional semantic network

CANARY(Tweety) WING(Tweety-left-wing)
PENGUIN(Opus) SWIMMING(Opus-swimming-style)
Part(Tweety, Tweety-left-wing)
Moves-by(Opus, Opus-swimming-style)

Besides the confused semantics of their relations, another deficiency of inheritance networks is that since information can only be represented about nodes, one cannot represent information about relations, such as that the `isa` relation is transitive. Nor can one represent information about beliefs, such as that the encyclopedia is the source of the belief that canaries are birds. Description logics do represent information about relations, but they do not represent information about beliefs. This deficiency is solved by propositional semantic networks, in which nodes are used to represent beliefs (propositions) as well as the individuals, categories, and properties represented by nodes in inheritence networks. Figure 2 illustrates a propositional semantic net in which `M1!` represents the proposition that canaries are birds, `M2!` represents the proposition that `isa` is a transitive relation, and `M3!` represents the proposition that the source of the proposition that canaries are birds is the encyclopedia.

7 Schemas, frames and scripts

Some researchers felt that semantic networks used a representation that was too fine-grained and too passive. Instead, they argued for representational structures that contain more information about the entities being represented, and that also incorporate active processes. They adapted the notion of schemas (or "schemata") from the psychological literature (*see* SCHEMAS IN PSYCHOLOGY). The two most widely used schema representation systems are frames and scripts.

Frames were originally proposed as a representation of structured visual information about complex objects. For example, if you open a door to an office, you expect to see certain things, such as a desk, chairs, etc. You would

Animal	
has-part:	head

Bird	
isa:	Animal
has-part:	wing
moves-by:	flying

Fish	
isa:	Animal
has-part:	fin
moves-by:	swimming

Canary	
isa:	Bird
can:	sing

Penguin	
isa:	Bird
moves-by:	swimming

Charlie	
instance:	Fish

Tweety	
instance:	Canary

Opus	
instance:	Penguin

Figure 3: Figure 1 as a frame system

be surprised to see a tennis court, a beach, and a swimming pool in the office. On the other hand, if you opened a door to a bedroom, you would expect to see a bed, a chest of drawers, etc. The proposal was that the “office frame” would contain pointers to the representations of objects you would expect to be in an office, the “bedroom frame” would contain pointers to the representation of objects you would expect to be in a bedroom, etc. As frame representation systems were implemented, they became more similar to semantic networks, but with the labelled arcs, now called “slots”, pushed into the nodes, now called “frames”, and the nodes pointed to by the arcs now called “slot fillers.” For example, Figure 3 shows the information of Figure 1 as a frame system.

One feature frame systems tend to have that semantic networks don’t is *procedural attachment*. Instead of a slot being filled by a pointer to a frame or a set of such pointers, the slot could be filled by an *if-needed* or an *if-added* procedure. If a slot containing an if-needed procedure is accessed, the procedure is executed and is expected to compute and return the slot filler. If a slot containing an if-added procedure is filled, the procedure is executed and is expected to fill other slots that depend on the new information being added to this slot. If-needed and if-added procedures are procedural versions of inference by backward chaining and forward chaining, respectively.

Scripts, like frame systems, were designed to be structured representations, but of activities rather than of objects. For example, the restaurant script contains a representation of all the activities that typically occur when one visits a restaurant. If you read a story about someone going to a restaurant and ordering a steak, you fill in the information about being seated and being given a menu from your restaurant script. (*See* MODELS OF ROGER SCHANK AND HIS STUDENTS.)

8 Pictorial representations

Some people think mostly linguistically; others think mostly in pictures. Everyone probably can do both, even though they usually do one or the other. Try this: Think of an elephant. Which way is it facing? If, when you thought of an elephant, you pictured one in your mind, you should have a definite answer to that question.

Just as people can represent entities in their minds either linguistically or pictorially (*see* MENTAL IMAGERY, PHILOSOPHICAL ISSUES ABOUT), we can use linguistic and pictorial representations in other media, including computers. The distinction is also often termed digital vs. analog, as in digital clocks vs. analog clocks.

The best way to distinguish analog from digital representations is to compare the domain of the representation (syntax) to the domain of what is represented (semantics). An analog representation has a syntactic operation that is a direct analog of a semantic representation. Consider clocks. What is represented is time. On an analog clock, the representation is the rotation of the clock hands around the circle of the clock face. The difference between the representation of 10:15 AM and that of 10:30 AM is a 90 degree rotation of the minute hand, which is one quarter of the complete 360 degree rotation. The complete 360 degree rotation represents one hour, and one quarter of a rotation represents one quarter of an hour. On a digital clock, however, the times 10:15 AM and 10:30 AM are represented with different numerals. Nothing about the difference between the two sets of numerals indicates what the difference in the represented times is, unless one moves to the separate semantic domain of numbers, and subtracts 15 from 30.

Analog representations can be constructed in computers by using a data structure whose operations are analogs of the relations being represented. For example, consider a predicate logic representation of items arranged in a row: *Left(desk, chair), Left(sofa, desk), Left(chair, stool),* where *Left(x, y)* means that x is to the left of y. To decide the left-to-right arrangement of the stool and the sofa requires a certain amount of search and inference. However, if, instead, the Left relation were represented by order in a list, the four relations would be captured by the list *(sofa, desk, chair, stool),* and the left-to-right arrangement of the stool and the sofa could be decided by a linear search. Some researchers have created systems that can reason about diagrams or visual scenes by representing them in two-dimensional data structures where it is easy to rotate, or otherwise manipulate the figures. (*See* ANALOGICAL REASONING, PSYCHOLOGY OF.)

9 Connectionist representations: local and distributed

Connectionist representations (*see* CONNECTIONISM; CONNECTIONISM, PHILOSOPHICAL ISSUES ABOUT) are designed to model the brain by using a large collection of intercommunicating units, each of which is a model of a neuron. These units are organized in layers: an input layer, an output layer, and one or more "hidden" layers. Each unit maintains an activation level and connections to other units, which may be on the same layer (in some versions) or on layers closer to the output layer. Each connection is also given some weight, which might be negative or positive. The network as a whole makes some decision or characterizes its input. Input is achieved by adjusting the activation level of the units in the input layer. When the activation of a unit exceeds some threshhold (which may be different for different units), an activation is passed to all outgoing connections, where it is adjusted by the connection weights, and passed to the connected units, etc. The final decision or characterization is read off the units in the output layer. Networks are trained by adjusting the weights on the connections by one of several possible feedback mechanisms. (*See* LEARNING IN CONNECTIONIST SYSTEMS.)

Local connectionist representations are distinguished by the requirement that each decision or characterization is represented by a single unit, and each input unit also represents some concept of the input. For example, in a lexical decision task, each input unit might represent a particular letter in a particular position, and each output unit might represent a particular word.

In a distributed connectionist representation, each represented decision or characterization is represented, not by a single unit, but by a pattern of unit activations. Distributed representations have been found to generalize what they have learned better than local representations do.

Connectionist representations are considered subsymbolic rather than symbolic representations (*see* CONNECTIONIST AND SYMBOLIC REPRESENTATIONS OF LANGUAGE; SYMBOLIC VS SUBSYMBOLIC). As such, they are not as capable of representing and reasoning about beliefs as the other representation techniques discussed in this article. (*Though see also* BAYESIAN BELIEF NETWORKS.)

10 Managing Change

Consider again some of the information in Figures 1 and 3, namely that birds fly, but penguins do not. If you learn that Opus is a bird, you are justified in concluding that Opus can fly. However, if you then learn that Opus is a penguin, you must reject your conclusion that Opus can fly. This is an example of an interesting phenomenon where a new piece of information causes the rejection of a previous conclusion. It is sometimes said that, in this case, the new piece of information *defeats* the old conclusion. This phenomenon often occurs in the presence of general information to which there are exceptions. The general information is sometimes referred to as *default* knowledge, and conclusions drawn from the general information are sometimes said to be `defeasible`. From the point of view of classical propositional and predicate logic, this situation is most unusual, since these logics are *monotonic*, meaning that if a conclusion can be drawn from some set of beliefs, it can also be drawn from any superset of those beliefs. (Just ignore the extra beliefs.) Attempts to formalize defeasible reasoning have been made by knowledge representation researchers (*see* NON-MONOTONIC LOGIC), and this remains an active area of research.

Removing default conclusions that have been defeated by more specific information is just one possible reason that information might have to be removed from a knowledge base. If the knowledge base is a model of the world, or a model of some agent's beliefs about the world, it may be that the world changes because of the action of the agent or some other agent. If the knowledge is a model of some agent, or a collection of beliefs input by some agent or agents, it may be that the agent or agents have changed their beliefs. If the knowledge base is a collection of facts and "laws" of some developing theory, it might be found that some of the facts and laws are contradictory, and the theory must be revised. Removing information from a knowledge base seldom involves merely removing a single proposition (fact, rule, law). If additional propositions have been derived from the one to be removed, they might need to be found and removed also. If the proposition to be removed was derived from other propositions in the knowledge base, or could be rederived from them, they must be found, and at least one of them must be removed or else the removed proposition could be reintroduced. The first systems knowledge representation researchers implemented to handle these complications of removing information from knowledge bases were called **Truth Maintenance Systems** (*see* TRUTH MAINTENANCE). More formal studies, carried out by computer scientists, logicians, and philosophers go under the name **belief revision**.

Using belief revision or truth maintenance to deal with a changing world is appropriate if the knowledge base always represents the *current* time, and should be changed as time moves. However, this eliminates the possibility of representing what was the case at past times. To do that, time must be represented explicitly, and propositions that hold only for some specific time must indicate so explicitly. To do this, specialized logics including temporal logics and modal logics have been used. Another logic for this purpose popular among knowledge representation researchers is the situation calculus, in which predicates that can change are given an extra argument that ranges over situations. For example, if a particular book is on a particular table in a particular situation, this might be represented as $On(\mathit{book1}, \mathit{table3}, \mathit{S5})$. In the situation calculus, an action is considered to be a function from the situation before the action is performed to the situation afterward. (Some versions of the situation calculus use slight variations of this.) For example, the action $\mathit{pickup}(\mathit{book1}, \mathit{S5})$ might represent the situation that exists after picking up *book1* in situation $S5$. We would then have $\neg On(\mathit{book1}, \mathit{table3}, \mathit{pickup}(\mathit{book1}, \mathit{S5}))$. Stating the effects of actions is rather straightforward. However, stating what is *not* changed by an action is more involved. For example, if it is the case that $In(\mathit{table3}, \mathit{room2}, \mathit{S5})$, is it the case that $In(\mathit{table3}, \mathit{room2}, \mathit{pickup}(\mathit{book1}, \mathit{S5}))$? The problem of specifying what is not changed by an action has been called the **frame problem** (*see* FRAME PROBLEM), not to be confused with the frames of Section 7, above.

References

Addanki, S. (1992). Connectionism. In Shapiro, S. C., editor, *Encyclopedia of Artificial Intelligence*, pages 268–274. John Wiley & Sons, New York, second edition.

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

Brachman, R. J. and Levesque, H. J., editors (1985). *Readings in Knowledge Representation*. Morgan Kaufmann, San Mateo, CA.

Cercone, N. and McCalla, G., editors (1987). *The Knowledge Frontier: Essays in the Representation of Knowledge*. Springer-Verlag, New York.

Davis, E. (1990). *Representations of Commonsense Knowledge*. Morgan Kaufmann, San Mateo, CA.

Hobbs, J. R. and Moore, R. C., editors (1985). *Formal Theories of the Commonsense World*. Ablex, Norwood, NJ.

Gärdenfors, P., editor (1992). *Belief Revision*. Cambridge U. Press, Cambridge, UK.

Iwańska, Ł. M. and Shapiro, S. C., editors (2000). *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*. AAAI Press/The MIT Press, Menlo Park, CA.

Kramer, B. and Mylopoulos, J. (1992). Knowledge representation. In Shapiro, S. C., editor, *Encyclopedia of Artificial Intelligence*, pages 743–759. John Wiley & Sons, New York, second edition.

Lehmann, F., editor (1992). *Semantic Networks in Artificial Intelligence*. Pergamon Press, Oxford.

Levesque H. J. and Lakemeyer G. (2000). *The Logic of Knowledge Bases*. MIT Press, Cambridge, MA.

Lifschitz, V., editor (1990). *Formalizing Common Sense: Papers by John McCarthy*. Ablex, Norwood, NJ.

Reichgelt, H. (1991). *Knowledge Representation: An AI Perspective*. Ablex, Norwood, NJ.

Rumelhart, D. E. and McClelland, J. L., editors (1986). *Parallel Distributed Processing*. MIT Press, Cambridge, MA. In two volumes.

Sowa, J. F., editor (1991). *Principles of Semantic Networks: Explorations in the Representation of Knowledge*. Morgan Kaufmann, Los Altos, CA.

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

Glossary

backward chaining A control structure used primarily in logic-based reasoning systems in which a question (or goal) is matched against an expression in consequent position of a non-atomic expression, or "rule", resulting in the expressions in the antecedent positions of that rule being generated as subquestions. The question is answered when all subquestions are answered by information in the knowledge base.

expert system A reasoning system that operates in a fairly narrow problem domain in which there are human experts. Such a domain is characterized by having problems with ill-defined solutions such that a proposed solution can be declared to be an actual solution only by an expert in the domain, and by having novices who become experts by an apprentice-like period of training at the end of which they are declared experts by the established experts.

forward chaining A control structure used primarily in logic-based reasoning systems and production systems in which new information is matched against expressions in antecedent (or pattern) positions of non-atomic expressions or rules, resulting in the expressions in the consequent positions of those rules being added to the knowledge base, being matched against other expressions in antecedent positions, etc.

graph, labeled directed acyclic A graph is a collection of vertices, or "nodes", and of arcs, each of which connects two nodes. In a labeled graph, the arcs are labeled by colors or symbols so that every arc is labeled and any label may be attached to one or more arcs. In a directed graph, the arcs are directed so that one might speak of an arc going from one node to another. A directed arc may be pictured as an arrow with the tail at the node it goes from, and its head at the node it goes to. In an acyclic graph, there is no path of arcs going from any node and eventually returning to it.

natural language Any of the languages learned by children as their native language, such as English, French, and Japanese, as opposed to formal languages, such as Algol and C, or invented languages such as Esperanto.

ontology In philosophy, the theory of what there is. In artificial intelligence, a knowledge base of information about a set of entities, especially one arranged in a taxonomy.