Knowledge representation in artificial intelligence.

Thomas John Evans

Follow this and additional works at: http://preserve.lehigh.edu/etd

Part of the Electrical and Computer Engineering Commons

Recommended Citation
KNOWLEDGE REPRESENTATION IN ARTIFICIAL INTELLIGENCE

by

Thomas John Evans III

A Thesis Presented to the Graduate Committee of Lehigh University in Candidacy for the Degree of Master of Science in Electrical Engineering

Lehigh University 1983
THIS THESIS IS ACCEPTED AND APPROVED IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE

Sept. 14 '43
(date)

Professor in Charge

Chairman of Department
# Table of Contents

Abstract 1

Introduction 2

1. Logic/Predicate Calculus 6
   1.1 Description 6
   1.2 Discussion 8

2. Semantic Networks 12
   2.1 Description 12
   2.2 Discussion 13
   2.3 Hybrid Systems 18

3. Frames 20
   3.1 Description 20
   3.2 Discussion 21
   3.3 Hybrid Systems 23

4. Procedural: Representations 31

5. Comparisons 36
   5.1 Aspects of Representation 36
   5.2 Representation of Knowledge 37
   5.3 Organization of Knowledge 40
   5.4 Inference Ability 42
   5.5 Learning Ability 46

6. A Complete Example 48

7. Representations for Problem Solving and Language Comprehension 63
   7.1 Description 63
   7.2 Discussion 66

8. Conclusions 70

References 72

Vita 73
## List of Figures

| Figure 2-1:     | Example of a semantic net - Cats | 12 |
| Figure 2-2:     | Network fragment - Smokey         | 14 |
| Figure 2-3:     | Example of a case frame           | 15 |
| Figure 3-1:     | Example of a frame - Cats         | 21 |
| Figure 3-2:     | Example of a frame - Ray          | 29 |
| Figure 5-1:     | Example of a semantic net - Clouds| 37 |
| Figure 5-2:     | Example of a frame - Clouds       | 38 |
| Figure 5-3:     | Network fragment - Morris         | 42 |
| Figure 5-4:     | Pattern fragment - To match Morris| 42 |
| Figure 6-1:     | Logic Representation - Initial    | 49 |
| Figure 6-2:     | Semantic Representation - Initial | 50 |
| Figure 6-3:     | Frame Representation - Initial    | 52 |
| Figure 6-4:     | Logic Representation - Story      | 54 |
| Figure 6-5:     | Semantic Representation - Story    | 55 |
| Figure 6-6:     | Frame Representation - Story      | 56 |
| Figure 6-7:     | Pattern Fragment - To match RABBIT| 58 |
| Figure 6-8:     | Pattern Fragment - To match CARROT| 61 |
| Figure 6-9:     | RABBIT1 Frame - Version I         | 61 |
| Figure 6-10:    | RABBIT1 Frame - Version II        | 62 |
Abstract

The organization of representations has become more complex. The first representation was logic, with very little organization, then semantic networks, and now frames. There seems to be a trend toward the more elaborate representations. The more complex organizations will make application of the intelligent processes easier. The ability to store information in small chunks will facilitate the problem solving process. The ability to group this information will facilitate the language comprehension process. I envision the representation of the future to be well structured, with many levels of organization. The lower levels will be more detailed than the upper levels. Each level will be individually accessible, so different intelligent processes can use different levels which are best suited for their particular processes. I would also suspect that certain concepts such as time relations, will be handled by separate representation systems.
Introduction

Webster's dictionary defines knowledge as the awareness of facts, truths, and principles. Simple facts are easy to represent, but truths and principles are more difficult. Typically, we think of knowledge in terms of facts about objects. The sky is blue. Dogs have four legs. These types of facts describe properties of objects. Winter comes before spring. Bob will be here tomorrow. These types of facts describe time relations between events. Events or actions are more difficult to represent because along with representing the event, a representation formalism may need to indicate the time relations between a series of events and their cause-and-effect relations. A behavior like riding a bicycle involves knowledge beyond that of objects and events, but knowledge about how to do things and the principles involved. There often is a thin line between this type of knowledge, called performance knowledge, and other types of knowledge. Then there is meta-knowledge, which is the knowledge about what we know. For example, we know that we do not know enough about knowledge representation.

There is more than just how knowledge is to be represented, but how it is organized, accessed, and processed. The actual use of knowledge involves three
principal stages: acquiring more knowledge, retrieving facts, and reasoning about these facts in search of a solution.

Knowledge acquisition or learning involves more than just adding new facts to a database. These facts need to be cataloged and checked for contradictions. If the program is told that all dogs have three legs, the program should not blindly accept this fact without first checking its database for contradictions. An intelligent program should answer, Dogs have four legs, not three. The program can acquire its information by being programmed, by being told, by studying examples, or by discovery.

To retrieve these facts, the program must determine which knowledge is relevant to a task and which is not. This is especially important when a system knows many things. The speed and efficiency of a system for solving problems is inversely proportional to the amount of knowledge it must process.

Reasoning is the ability to solve problems which it has not been explicitly told how to do. There are five different types of reasoning: formal, procedural, by analogy, generalization, and meta-level reasoning. Formal reasoning involves the syntactic manipulation of data structures to deduce new ones following prespecified
rules of inference. Procedural reasoning uses simulation to answer questions and solve problems. Reasoning by analogy and generalization are very natural for humans but quite difficult to accomplish in artificial intelligence programs. Meta-leveling reasoning involves knowledge about what you know, and may play a key role in human cognitive processing [1].

A closely related field is psychology, which studies how the human brain works. Many representations are modeled after psychological models of the brain. It is logical to assume that if we are to duplicate the performance of the brain, then we must simulate its functioning process. The human brain is separated into different sections for different functions. There are two speech sections. One section is sandwiched between the auditory area and the visual area and is responsible for understanding words. I don't think it's an accident that it's between the auditory area (spoken words) and the visual area (written words). The other speech area is responsible for speaking words. Although these areas are separated, they can communicate to each other by neurons traveling from one section to the other [2].

Should the field of artificial intelligence follow this example, and have different systems for different functions with ability to communicate with each other?
There is no simple solution, as can be seen from a recent SIGART newsletter, "Special Issue on Knowledge Representation" [3]. This consisted of answers to an elaborate questionnaire on various topics in knowledge representation. The result was a wide diversity of answers and opinions. There is no concrete mathematics to follow; artificial intelligence is more of a trial and error science. Only hard work and further research will answer these questions.
1. Logic/Predicate Calculus

1.1 Description

One of the first representations developed was logic, also known as predicate calculus. Logic is a very formal representation, developed by mathematicians, it is similar to Boolean algebra. It is concerned with the form or syntax of statements and with the determination of truth by syntactic manipulation of formulas. The best known inference rule is modus ponens. It states that if we know that two sentences of the form $X$ and $X \rightarrow Y$ are true, then we can infer that the sentence $Y$ is true. For example, if we know that the sentence John is an uncle is true and we also know that If John is an uncle, then John is a man is true, then we can conclude that John is a man is true. More formally, the modus ponens rule would be expressed as:

$$(X \land (X \rightarrow Y)) \rightarrow Y$$

Where $X \rightarrow Y$ means that the truth of $X$ implies that $Y$ is true.

The difference between logic and predicate calculus is that predicate calculus is an extension of the notions of logic. Predicates were added to the logic to simplify the representation. Predicates are statements about
individuals, both by themselves and in relation to other individuals. A predicate is applied to a specific number of arguments and has a value of either true or false. Predicate calculus systems usually have functions, which are like predicates, but return not just the values true or false, but return objects related to their arguments as well.

Predicate calculus is an extension of the notions of logic. The meanings of the connectives (i.e. \(\land, \lor, \rightarrow\)) are retained, but the focus of the logic is changed. Instead of looking at sentences that are of interest merely for their truth value, predicate calculus is used to represent statements about specific objects, or individuals. For example, the English sentence All carrots are vegetables, is thus expressed in predicate calculus as:

\[
\forall x. \text{CARROT}(x) \rightarrow \text{VEGETABLE}(x)
\]

Logic and predicate calculus are often used interchangeably. Most present systems used what is actually called first order predicate calculus or first order logic. A logic is of first order if it permits quantification over individuals but not over predicates or functions.
1.2 Discussion

Logic's strong point is that it is mathematical in nature. Its rules of inference are well understood and easy to apply. It is most commonly used in theorem proving, or used in combination with other representations. Logic is very precise, it has a clean syntax and clear semantics. Logic is also modular, new facts can be entered in a database independently of each other. Logic is also flexible, since it makes no commitment to the kinds of processes that actually make deductions, a fact is represented without having to consider its possible use.

While logic seems a natural way to express certain notions in problem solving, it is quite inadequate for language comprehension. Logic can't represent time relations, fuzzy concepts, and has no way of grouping information. Nor does it have a way of controlling inferences, which would be a problem with a large database. The most common complaint about predicate calculus is that it does not have a good organization [3]. Its method of accessing and storing information is poor, it ignores aspects of knowledge representation such as memory access and organization.

Charniak discusses the inference ability of a number of systems, and the problem of how knowledge is used to
make inferences in the comprehension of language [4]. Charniak proposes that a complete solution would require answering the following questions:

1. **Semantic Representation.** What concepts, and in what combinations are needed to record our impressions of the world?

2. **Inference Triggering.** Under what circumstances, and for what reasons do we make inferences?

3. **Organization.** Given we want to make an inference, how do we locate the needed information?

4. **Inference Mechanism.** Once we have located a fact, how do we know how to use it?

5. **Content.** What is the knowledge which we have of the world that enables us to understand language?

The ideal representation would be able to answer the five questions, and if it could answer these questions, it would be able to understand language. What would the answers be? First, it would give us a basic list of primitives to which all concepts could be translated, similar to translating a high level programming language to assembly language. Once the knowledge has been translated into primitives, the processing of those primitives would be much easier because the concepts would be in a condensed version. This would make pattern
Hatching much easier, along with making the database smaller and easier to handle. Second, it would make inferences continuously, and it would add the newly discovered information to its knowledge base. It would also be able to selectively filter out unwanted theorems and know where to look for the needed information. It would also know how to apply a fact, once the fact has been found. Lastly, it would know what is the knowledge which we have of the world that enables us to understand language.

To answer the first question, predicate calculus does not tell us what predicates are needed to represent our impressions of the world, in other words, what are the necessary primitive predicates. For inference triggering, predicate calculus only makes inferences when asked a question. The organization does not help filter out useless data, the inferences must be made on the entire database. As mentioned before, predicate calculus has no way of controlling its inferences, and therefore fails miserably with large databases. Its main strong point is that its inference mechanism is excellent, which is why it is used in theorem proving. To answer the last question, predicate calculus says nothing about what facts are needed. As it turns out, some facts are difficult, if not impossible, to express in predicate
Newell discusses the controversy of predicate calculus which developed in the past in his paper on the Knowledge Level [5]. After early work it became clear that logic alone was not the answer, especially in the areas of robot planning, language comprehension, and some problem solving. The residue that resulted was that logic is a bad representation. Newell points out that logic is an appropriate tool for analyzing the knowledge, though it is often not preferred to be the representation itself. Newell further states that to determine exactly what knowledge is in the representation and to characterize it requires the use of some form of logic. This is why predicate calculus is commonly used in combination with other systems.
2. Semantic Networks

2.1 Description

Semantic networks, sometimes called semantic nets, are representational formalisms consisting of nodes and of links connecting the nodes. Both nodes and links can have labels. Nodes usually represent objects, concepts, or situations in the domain, and the links represent the relations between them.

If we wanted to represent the fact that all cats are mammals in a semantic network, for example, we would create two nodes to designate cats and mammals with a link between them. If we wanted to add the fact that all mammals are warmblooded, this would be done by creating another node to designate the characteristic of being warmblooded, linked to the node designating mammals. These links are commonly called ISA links. Graphically, the representation would look like this:

```
| CATS | --- > | MAMMALS | --- > | WARMBLOODED |
```

**Figure 2-1:** Example of a semantic net - Cats

Semantic networks are often explained with graphic examples. A reasonably elaborate network would look extremely complicated.
2.2 Discussion

We can easily see from this representation that cats are warmblooded. The ease with which it is possible to make deductions about inheritance hierarchies is one reason for the popularity of semantic networks as a representation. In a domain where much of the reasoning is based on a very complicated taxonomy, a semantic network is a natural representation scheme. Semantic networks can represent all types of relationships by appropriately naming the links.

A drawback of links is that, by their nature, can encode only binary relations. What is needed is the semantic net equivalent of a multi-variable predicate in logic. To solve this problem, we allow nodes to represent situations and actions, as well as objects and sets of objects. Each situation node can have a set of outgoing arcs, called a case frame, which specifies the various arguments to the situation predicate.
For example, if a cat named Smokey owns a toy, the semantic representation would look like this:

```
| SMOKEY |--isa-->| CAT |--isa-->| MAMMAL |

|          | |--isa-->| TOY |
```

Figure 2-2: Network fragment - Smokey

where TOY1 is a specific instance of TOY. Now suppose we wanted to represent the fact that Smokey owned TOY1 only from Halloween to Christmas. This means we would need the start-time and end-time of the act of ownership. To do this, a node called OWN1 would be created to represent a particular case of ownership. It would inherit case arcs to the owner, ownee, start-time, and end-time.
The representation would then look like this:

```
| SMOKEY  |--isa--> | CAT  |--isa--> | MAMMAL |
|         | O         |          |         |
|         | owner     |          |         |
|         |           |          |         |
| OWN1  |--ownee--> | TOY1  |--isa--> | TOY |
|        |           |          |         |
|        | start-time--> | HALLOWEEN |--isa-->
|        |           |          |         |
|        | end-time--> | CHRISTMAS |--isa--> | TIME |
|        |           |          |         |
|        | isa--> | OWNERSHIP |--isa--> | SITUATION |
```

Figure 2-3: Example of a case frame

For certain actions, default values are associated with the case arcs. For example, if we wanted to represent the fact that Smokey sleeps, default values would be given for the start-time and end-time giving the times cats usually sleep. It also may be one value, such as a period of the day, such as night, day, afternoon, and so on.

This representation tends to lend itself to the
expression of states and actions in terms of a small number of primitives concepts. The use of a small number of semantic primitives as the basis of a system's knowledge representation has both advantages and disadvantages. The ability to translate all concepts and ideas down to a few primitives has the advantage that it would make processing much easier. By processing I mean such things as pattern matching, inference making, etc. However, it has not been proven that any particular set of primitives are sufficient to represent the meaning of every English verb.

A problem with semantic networks is that there is as yet no distinction in our network formalism between an individual and a class of individuals. Some things said about a class are meant to be true of all members of a class, like cats are mammals, while some refer to the class itself, for example, cats are an endangered species. If cats are an endangered species, this means cats in general. A particular member of the class would not necessarily be considered endangered.

In semantic network representations, there are no formal semantics, no agreed upon notion of what a given representational structure means, as there is in logic. Meaning is assigned to a network structure only by the nature of the procedures that manipulate the network.
The reasoning mechanism used by most semantic network systems is based on matching network structures. The matcher can make inferences during the matching process to create a network structure that is not explicitly present in the network.

There are other problems, common to other representations. One such problem is the computational problems that arise when databases become large. Others are, what does a node mean? Is there a unique representation for an idea? How are time relations to be represented? How are ideas and beliefs represented? What are the rules of inheritance [1]?

Semantic networks are a very popular representation scheme in artificial intelligence. Most work on the representation of knowledge involves elaboration of the semantic net idea, in particular, work on aggregate network structures called frames.
2.3 Hybrid Systems

The systems discussed so far are designed and developed with the present day common computer systems. As discussed in the introduction, the brain is separated into different areas for different functions, these areas performing their functions simultaneously. This gives us the ability to do many things at one time. Today's computer doesn't perform like the brain, it performs only one action at a time, in series.

Fahlman proposes a representation which utilizes its own hardware system, designed to imitate a psychological model of the brain [6]. NETL consists of two parts: the parallel network memory scheme, and the vocabulary of conventions. The parallel network is a semantic network representation with the ability to access many nodes simultaneously. It works by injecting markers into the network, and letting them propagate through the network. This network will be built with the appropriate hardware necessary to control and access all the nodes. The second part is the vocabulary of conventions, which are developed to be simple, but yet be fully functional. The hardware that is necessary has not been built, so NETL remains only a theoretical system.

If it is ever built, this system will have the advantage of being able to perform very fast searches.
It will be able to handle large databases better than normal computers. But because it is a semantic network representation, it still has many of the problems that other network representations have. For example, it would need a consistent set of primitives and formal semantics. Will it also be able to reason by analogy or by generalization?
3. Frames

3.1 Description

A frame is a data structure for representing a stereotyped event or object. Frames are commonly thought of as nodes and relations. The top levels of a frame are fixed and represent things that are always true about the supposed situation. The lower levels have many slots that must be filled by specific instances or data. Each slot can specify conditions its assignments must meet. A frame’s slots are normally already filled with default assignments. It may also have a range attached to a slot which would indicate what information is considered valid for that slot. The default assignments can be easily replaced by new items that better fit the current situation. The frame systems are linked by an information-retrieval network [7]. Consider an example of a frame representing a cat.

```
CATS[
  ISA:MAMMAL,
  VISION:EXEMPLARY,
  COLOR:BLACK
  ;RANGE(BLACK,WHITE,GRAY,BROWN)
  AGILITY:EXEMPLARY
  INTELLIGENCE:GOOD
]
```

Figure 3-1: Example of a frame - Cats
3.2 Discussion

Frames are popular because they group information into subsets. This makes it easier to access and process pertinent information. Once a frame has been accessed, it becomes active, and all the slots are then accessible. Slots contain important information that is sure to be needed for any intelligent process. One such process is the comparing of events, actions, and other things which are represented by frames.

Frames were developed from a psychological model of the brain. According to this model, the brain brings blocks of information into short term memory for use. Frames are still quite new, and more research is needed. The one criticism of frames is they wouldn’t allow for deep inferences which are needed for complex problem solving. This would be true if the information-retrieval network was not set up properly. Given that a system can filter out unwanted information without filtering out needed information, this shouldn’t be a problem. I think the problem is to insure that the inference making is on the right track in the first place. This is especially important in making inferences on imprecise knowledge. For example, let’s take the case of medical diagnosis. The computer may be given an initial set of body function measurements, from which it infers that the patient has
some sort of virus infection. After making more tests, it finds that the patient doesn't have an infection, it then has to realize that it made the wrong inference, and try another approach.

In logic, inferences are made with the entire database. This would be analogous to making every possible medical test on a patient, not an acceptable solution. Also it is important to note that the system may have to make inferences about its own knowledge, to check for mistakes. Logic's rigidity does not allow for fuzzy logic, with frames it is still unknown.
3.3 Hybrid Systems

The frame theory is considered to be quite new and many questions need to be answered. One attempt to use the frame theory in language comprehension was made by Charniak [8]. Frames are used in language comprehension because of the natural way they partition knowledge. One such system, developed by Charniak, is able to understand simple stories about painting. Charniak's system has a number of interesting features and modifications to the frame theory.

One such modification is the comes-from and leads-to links. The comes-from links indicate a stereotyped way of achieving a given goal. For example, the stereotyped way of cleaning a brush is by washing it with an appropriate solvent, not by dry cleaning. Leads-to links indicate why an action is performed. A leads-to link would indicate that the purpose of washing a brush is to remove the paint.

This system also has a simple deductive inference program, which recognizes three types of inferences: consistency, unexpected situation, and needed fact inferences. Consistency inferences are made when a new fact contradicts an old one. This happens when a previous inference jumped to an incorrect conclusion. Unexpected inferences are made when one of the frame's
expectations is confounded. Needed-fact inferences are used to indicate which facts are important to the particular situation.

This system uses five types of frames: complex-event, simple-event, state, adjunct, and object frames. A complex-event frame is one for which the primary connection between sub-events is temporal. There are no causal relations in a complex-event frame, these are expressed in simple-event frames and attached through a leads-to link. This restriction tends to enforce a certain type of modularity on the representation. A state frame describes a state much like the way an event frame describes an action. Object frames describe physical characteristics of objects along with their typical uses. The last type of frame is the adjunct frame, which is also used to maintain modularity, which is used to organize multiple actions within a frame.

This program has a number of problems which have yet to be solved. One problem is that it can't distinguish between (not(cause x)) and (cause(not x)). Another problem which is common to most representations is the inability of the system to handle time relations. The system has no backup, so all input must be in chronological order. For example, the system couldn't handle the input: Bob's birthday is after Tom's, but
These are specific problems. A more general problem is that of the search. The search problem is, in essence, where shall it go when looking for frame statements which match the input. The system only looks in currently active complex-event frames. The ability to search intelligently is very important to the system's ability to make inferences. Not only should the system search currently active complex-event frames, but should search its siblings as well. As the size of the representation grows, the deeper and more complex this kind of search will become. This is still another problem, the inability to search selectively which would give the system the capability to handle more than one subject. As of now, the system can only handle the subject of painting.

This system also has a number of problems with the representation itself. One such problem is the inability to represent "had to" or "could", but this problem is very common. This system also has no way to represent "where" questions, because it does not currently have the concept of location.

Charniak's representation stresses modularity, however the use of leads-to and comes-from pointers to individual frame statements make it quite difficult to
write or read frames since they are constantly referring to the details of many other frames. Another problem with the pointers is the use of intermediaries, which were introduced in order to allow a comes-from or leads-to links to point to something which was not a direct match, but which can be made to match by applying a certain rule. This introduced a new and unwanted degree of freedom in the construction of frames.

It needs to be stressed that the representation was put in manually. There is no learning process by which new frames can created, due in part to the difficulty in reading and writing frames. Also, the input was not in the form of English sentences, but in semantic representations.

Although this program has its share of problems, it does attack some common problems in artificial intelligence. Probably the major force in the design of the frame representation is the goal of modularity. The restrictions on complex-event frames and the use of adjunct frames allowing special cases to be represented separately facilitate modularity and cleanliness. The frame representation also keeps a fairly clear separation between facts it knows and information about how these facts are to be used. This system represents states and goal states. This would enable it to be used with a
problem solving system, to be able to find paths to goal states.

The ability to learn, without being explicitly being told or programmed, is an intelligent quality that humans have. One idea is to use a theory of creating transfer frames to learn through simile-like examples. For example, when we hear Ray is like a cat, we make the use of what we know about cats to transpose the qualities of cats into our conceptual idea of Ray. If we were previously talking about Ray's athletic ability, we would then deduce that Ray is very agile. If the previous reference was about eyesight, we would deduce that Ray has good vision. Winston proposes an idea of transferring frames to give frame representations this ability [9].

In Winston's system the teacher's simile determines a destination frame and a source frame. The system, on analyzing the sentence, computes a transfer frame. It stands between the source and destination, acting as a filter, to determine exactly what slot-value combinations are allowed to pass from one to the other. Computing the transfer frame requires two steps: hypothesis and filtering. In the hypothesis step, potentially useful transfer frames are produced through an analysis of information in the source frame and its immediate
relatives. In the filtering step, the better of the hypothesized frames are selected through a study of the destination frame and its relatives, together with the things learned in previous instructions.

The hypothesis methods concentrate on the source and its context. The methods, briefly summarized, are:

1. Use a remembered transfer frame.
2. Make a transfer frame using slots with extreme values.
3. Use slots that are known to be important.
4. Use slots that have unique values with respect to their siblings.
5. Use all of the slots.

The filtering methods focus on the destination frame and its context. They are as follows:

1. Prefer transfer frames that have slots that are present in the typical instance associated with the destination.
2. Prefer those that have slots that some sibling of the destination exhibits.
3. Prefer those that are in the same property group that was involved in the last transfer.

Given the example: Ray is a person. Ray is on the baseball team. Ray is like a cat. The system would first create a frame for Ray and would fill in the slots from
the stereotype values in the person frame. It would then add a link to indicate that Ray is a member of a baseball team. The third sentence would then trigger Winston's intelligent mechanism. Using method 2 for hypothesis of transfer frames, we get two frames, one for vision, and one for agility. Using method 3 for filtering of transfer frames, we would filter out vision since the last topic was athletics. We could then deduce that Ray is very agile, and would add this to his frame. Winston does not discuss language comprehension, he only discusses processes used in learning by comparison. This mechanism would be triggered by phrases such as like, unlike, is the opposite, etc.

Although his system worked well for a few examples, this theory of Winston's was only tested with the small blocks world. The number of slots is small, and they are carefully arranged. What would happen as the size of a representation increases and becomes more complex? Winston states that the frame representation will tend to keep information under control. However, only further
research will answer these questions. Like people, this system works well when examples are clear and precise. If comparisons are inconsistent or contain too many properties, the system would be confused and would make the wrong deductions.
4. Procedural Representations

Most representations discussed in this paper were implemented in the LISP programming language. Some argue that a better language is needed, one that is more suited to needs of inference making. Many representations concentrate on representing declarative knowledge, as opposed to procedural knowledge. Declarative representations stress the static aspects of knowledge; facts about objects, events, and their relations with each other and states of the world. Procedural representations stress dynamic aspects of knowledge; knowing how to use their knowledge, how to find relevant facts, make inferences, and so on.

Procedural representations are often used in expert systems. They are also used in conjunction with other representation schemes. For example, predicates used in predicate calculus are a form of procedural representation. As in logic, they are used to specify control and perform other actions involving the knowledge base. They also can be used to handle special cases of knowledge representation, such as representing time relations.

One such language designed, but never implemented, was PLANNER. PLANNER was an artificial intelligence language designed to implement both representational and
control information. The specific concern of PLANNER was not to facilitate the class of inferences that were logically possible, as would be the focus of theorem proving work, but to expedite the inferences that were expected to be actually needed.

The relevant features of the PLANNER language included being able to specify whether theorems should be used in a forward or backward direction, and the ability to recommend specific theorems, or even an entire class of theorems. PLANNER thus serves as a programming language in which knowledge about both the problem to be solved and the methods of solution can be stated in a modular, flexible style reminiscent of logic.

Two specific criticisms have been directed at PLANNER's method of specifying control. First, it is too local, thus it is unable to consider the overall shape of the problem's solution. Second, it cannot reason about its control information.

Chief among the advantages of using procedures to represent knowledge is their facility for representing heuristic knowledge. A related advantage is the ability to perform extended logical inferences. Another advantage is its ability to model side effects of actions taken in the world by making corresponding modifications in the database representing the state of the world. The
procedure that performs the actions can update the database immediately.

Two general problems of the procedural approach in relation to more formal representational schemes concern completeness and consistency. Many procedural systems are not complete, meaning there are cases in which a system, like PLANNER, could know all the facts required to reach a certain conclusion, but not be able to make the required deductions. Also the use of default reasoning can introduce inconsistency in the presence of incomplete knowledge. Because of the interaction between various facts is unavoidable due to the heuristic information itself, modularity is sacrificed [1].

What advantages does PLANNER have over LISP as a language for making inferences? First, database management is made easier by the use of functions such as assert, goal: and erase. Secondly, PLANNER offers a more sophisticated pattern matching facility. Third, theorems could be called on the bases of their pattern. This would enable functions to call other functions without ever knowing the name of the secondary function. Finally, PLANNER has the ability to backtrack, but there must be a way to turn if off if it is to be an advantage. Sometimes the search is not a blind one, in this case, backtracking would not be desirable [4].
How does PLANNER compare to first order predicate calculus? For one, PLANNER allows the user to specify how a goal or theorem is to be established. This gives it the ability to control inferences, thus it suffers less from combinatorial explosion. Predicate calculus does not have this capability. Predicate calculus also has a problem with coping with contradictions. This is no problem in PLANNER, since you write your own theorems, you would have explicitly to write a theorem which would derive something from a contradiction before that would happen. Also, data-driven inferences are easier to accommodate than in predicate calculus.

What does PLANNER say in response to the five questions ask by Charniak [4], as discussed in chapter 1. PLANNER says nothing about what you can put in a assertion. It makes no restrictions about when to make inferences. PLANNER does tell us something about how to locate the needed information, but in a very qualified way. The primary one is pattern directed invocation. The secondary one is the means by which one can choose which theorems are to be used to satisfy a goal. However, PLANNER is a programming language, and therefore it is possible to program in other organizations. Like in predicate calculus, PLANNER is primarily a theory of inference mechanism. PLANNER, also says nothing about
the language content.

As mentioned before, procedural representations are commonly used in expert systems. They concentrate on one idea, they don’t have to worry about the problem with large heterogeneous databases. Although they are limited in their topics, they do exhibit quite intelligent behavior. There are a number of expert systems working, most using the concepts of pattern invoked programs and procedural representation [10].
5. Comparisons

5.1 Aspects of Representation

Each representation has its disadvantages and advantages, no representation is perfect. Some representations are better suited for language comprehension, some for problem solving. The question that is often asked is, which representation is the best?

I will divide the aspects of the representation into four distinct parts:

1. Representation of Knowledge. Can the representation fully represent all the knowledge in the world?

2. Organization of Knowledge. How does the organization help the artificial intelligence process?

3. Inference Ability. How does the representation handle inference making?

4. Learning Ability. Does the system have the ability to learn?

A perfect representation will be able to represent all knowledge, it will be able to organize it efficiently, it will be able to make inferences effectively, and it will be able to learn without having to be programmed.
5.2 Representation of Knowledge

Can the representation fully represent all the knowledge in the world? All representations discussed so far are capable of representing declarative facts, like the block is red and blue. Declarative facts are easy to represent because there are no complex relations involved, and usually no ambiguity. How would one represent the fact that when it is cloudy, it sometimes rains? This type of fact is not in a black and white area, but it is in a gray area. Sentences that include the words might, sometimes, maybe, and so on are relating actions and events by means of fuzzy relations. In logic it probably would be represented by an OR operator, cloud $> \text{rain v norain}$. This would probably be fine for problem solving, but you would probably want to treat the phrase rain $v$ norain as a separate entity in language comprehension. In semantic networks, the representation would look something like this:

```
| CLOUDS |--want--> | RAIN |
```

Figure 5-1: Example of a semantic net - Clouds

The want link would represent that the relationship between the clouds and the rain does not always happen. Remember that these links are unidirectional, so the
relationship can't be reversed. The fact that if it rains, it must be cloudy does not imply the fact that if it is cloudy, it must be raining. Frame representation would be similar, and would look like this:

```
CLOUDS [
IS:SUSPENDED-WATER
COLOR:OPAQUE
CAUSE:EVAPORATED-WATER
RESULT:RAIN
]
```

Figure 5-2: Example of a frame - Clouds

where the question mark indicates that rain may or may not occur.

One can also see from this example that representing complex objects and events is more difficult. When something such as a cat is to be represented, its complete description can be developed by following the ISA links upward to the parent frames and nodes in semantic nets and frames. With something such as clouds, which has no hierarchical system, a complete description of a cloud is difficult to understand, and often unnecessary. Representation of a cloud in semantic networks would require an entire network of many nodes, and depending on the application, may not be necessary.

Logic has no way of representing time relations. Semantic nets and frames can represent basic relations but not complicated ones. A basic time relation would
simply indicate which event came first, which came second, and so on. A more complicated time relation would require backtracking and a sophisticated procedure to sort the various events. Semantic nets and frames can represent the relations in a crude way, but are unable to use the representation. Each node would represent a particular action and would be linked to other actions by links indicating the time relation between them.
5.3 Organization of Knowledge

How does the organization help the artificial intelligence process? This is probably logic's greatest weakness. Logic has no way of grouping information, thus it can't sort large data bases into a useful format. Knowledge is stored in small individual chunks, which is good for problem solving, but not language comprehension. However, even in problem solving, there must be some way to quickly filter out unrelated information.

Semantic networks group the information inherently. A separate cross-reference dictionary would give the location of all nodes by their type. For example, if the topic of schools were introduced, the system could locate all the school nodes in the network. Some of these nodes may not be applicable, for example, one of the school nodes may be linked to a fish network, but at least the information needed to be processed is greatly diminished. Once a particular node is located, it is then linked to related information, and can be searched outward to locate the needed information.

Frame representations group the information in two ways. Like semantic networks, they are organized by a network of links, which can be located with the help of a cross-reference dictionary. But they also group the information into frames. Given a topic which is
represented by a frame, certain information is stored in the frame, and can be quickly accessed. By grouping knowledge about a topic in this way, various specific instances of this topic can be compared and processed easily and quickly. In semantic networks, a pattern matching program would be used, which can be time consuming. Also by grouping information into such subsets, adding and deleting groups of information become easier than having to trace and delete an entire network in a semantic net representation, for example. Because of their excellent organization of information, frame representations have become very popular in artificial intelligence.
5.4 Inference Ability

How does the representation handle inference making? Logic’s strength is that its rules of inference are well understood and easy to apply. Given that its knowledge is true, then any inferences made are guaranteed to be true. Logic is this way because it is based on mathematical principles.

A semantic net uses a pattern matching process. It constructs a pattern which it seeks to match, and then searches for a similar match. Note that it must be able to match similar nets as well as exact ones. For example, given that we have the following network fragment:

```
| CATS | <--isa--| MORRIS |--does--> | TV-ADS |
```

Figure 5-3: Network fragment - Morris

Now if the system is asked, is there a cat that performs television commercials, the pattern matcher constructs a pattern like this:

```
| CATS | <--isa--| ? |--does--> | TV-ADS |
```

Figure 5-4: Pattern fragment - To match Morris

The matcher would then have to search not only for cat
nodes, but would trace these nodes to specific instances of cats by way of ISA links. It can easily be seen that this is quite a complicated process. Note how important it is to have a standardization of names for knowledge concepts. Synonyms must be translated into standard primitives. The frame's inference mechanism would be similar, it would check specific instances of cat frames for slots indicating that it does commercials.

Logic is well designed for formal reasoning, but doesn't seem well constructed for the other types of reasoning as discussed in the introduction. Semantic networks use a pattern matching scheme to give them their reasoning power. This type of reasoning can best be classified as procedural, because the system simulates the knowledge which it needs. Semantic networks also do not seem well constructed for the other types of reasoning.

Frames seem better suited for other types of reasoning than the previous methods. As discussed in 3.3, frames seem well designed for reasoning by analogy and generalization. Although frame systems are still in the experimental stage, they are nevertheless a big step further than the other representations. By using logic in combination with a frame representation, you can give it the power of formal reasoning. It is easier to
incorporate logic with frames than with semantic nets because frames are better constructed for complex organization.

Let's look at an example of a more complicated reasoning process. Given the story fragment:

Jack was building a fence. He was almost finished when his hammer broke. Could he use a large rock as a hammer? Could he use a large marshmallow as a hammer?

This is often referred to as commonsense knowledge. The reasoning process would go something like this.

A hammer is heavy and hard. A large rock is heavy and hard. So a rock could be used as a hammer. A marshmallow is soft and light, so it could not be used as a hammer.

Ignoring the fact that some rocks are soft, and they might chip, and so on, we can see how we are comparing the various attributes the hammer and its various substitutes. This type of reasoning requires the ability to compare objects.

In logic, a theorem would be needed, stating the fact that if two objects have similar characteristics, they can be used interchangeably. But this is not always correct. A large glass vase is hard and heavy, but will shatter if it is used as a hammer. So what would be needed is a theorem for each particular case of
replacement. For semantic nets and frames, a system would search for characteristic links. It would have to check through all links or slots to check for undesirable characteristics.

To be able to make such decisions, a system would need more than one example. When given more than one example, it can check each example for common characteristics from which it can conclude that all objects with the same characteristics will perform the same task. One can easily see the amount of information that is needed to be exposed to and stored is enormous, even for something as simple as the example of a hammer. The ability to store and search through large amounts of information quickly is a basic ability a system must have if it is to be intelligent.
5.5 Learning Ability

Does the system have the ability to learn? I define learning as the acquisition of knowledge, as opposed to the sub-ability of reasoning, which I define as the ability to infer and deduce new facts from this knowledge.

The simplest type of learning is when a system is given a new piece of information. The system must have the ability to organize and construct its own database. In many of the proposed systems discussed in this paper, the representation was constructed by hand. Because of the problem of constructing links and translation of concepts into primitives, frame and semantic network systems are quite difficult. Logic is easier because of the independence of the individual pieces of knowledge. Also, logic does not need primitives, for as long as synonymous relations are known, the inference mechanism will make the appropriate deductions.

Another type of learning is when the system makes inferences to discover new facts. A system would want to store new knowledge acquired from the various intelligent processes. This knowledge must not be considered permanent, so that if it is found incorrect, it can be changed. However, some facts you don't want to change. For example, the fact \#hours/day = 24 should not be
easily changed. A system should inquire about new facts that contradict old facts. As discussed before, most systems only make inferences when asked a question. This means a system will have to make inferences whenever new information is given. For example, when new frames are created, values are placed in the slots of the frame. These values can easily be replaced, but the links to other frames are usually considered permanent.
6. A Complete Example

Now let's illustrate the various representations with a complete example. Given the following story:

John and Jane grow vegetables in their garden. This morning he saw a brown or orange rabbit eating in the garden. He can't see the difference because he is colorblind. He decided to build a fence around the garden. He was almost finished building the fence, when his hammer broke.

How would this story be represented?

Each representation would need some basic knowledge, given here in English:

1. Carrots are vegetables.
2. Vegetables are plants.
3. A garden is used to grow plants.
4. Rabbits are mammals.
5. The color of a rabbit is black, brown, or white.
6. Rabbits eat carrots.
7. A fence is used to keep something in or to keep something out.
8. A hammer is used to hit something.
9. A hammer is heavy and hard.
10. Rocks are heavy and hard.
The predicate calculus representation of the above knowledge would look something like this:

\[
\begin{align*}
&\forall X. \text{CARROT}(X) \rightarrow \text{VEGETABLE}(X) \\
&\forall X. \text{VEGETABLE}(X) \rightarrow \text{PLANT}(X) \\
&\exists X. \text{GROW}(X) \\
&\forall X. \text{RABBIT}(X) \rightarrow \text{MAMMAL}(X) \\
&\exists X. \text{COLOR-RABBIT}(X) \\
&\exists X. \text{EAT}(X) \\
&\exists X. \text{KEEPIN}(X) \\
&\exists X. \text{KEEPOUT}(X) \\
&\exists X. \text{HIT}(X) \\
&\forall X. \text{HAMMER}(X) \rightarrow \text{HEAVY}(X) \land \text{HARD}(X) \\
&\forall X. \text{ROCK}(X) \rightarrow \text{HEAVY}(X) \land \text{HARD}(X)
\end{align*}
\]

Figure 6-1: Logic Representation - Initial

where the V stands for the Boolean function OR, and the \(\land\) stands for the Boolean function AND. There are three types of statements shown: a simple expression, a predicate, and a function. A simple expression is of the form \(\forall X.----\), which is loosely translated as: for all \(X\). The first statement represents the fact that if \(X\) is a carrot, then \(X\) must be a vegetable. A predicate is of the form \(\exists X.----\). The predicate \text{COLOR-RABBIT} will return the value TRUE if \(X\) is black, brown, or white. The predicate \text{HIT} will return the value TRUE if \(X\) is a \text{HAMMER}. A function is of the form \(\exists X.----\). The function \text{GROW} will return the value \text{PLANT} if the value of \(X\) is \text{GARDEN}. The functions \text{KEEPIN} and \text{KEEPOUT} will return the corresponding values if \(X\) is a \text{FENCE}, these values are not known yet.
The semantic representation of knowledge shown previously would look something like this:

--- CARROT ----isa--> --- VEGETABLE ----isa--> --- PLANT ---

--- color --> --- BLACKvBROWNvWHITE ---

--- RABBIT ----isa--> --- MAMMAL ---

--- keep-in --> --- FENCE ---

--- keep-out --> ---

--- hit --> --- HAMMER ---

--- prop --> --- HEAVY^HARD ---

--- prop --> --- HEAVY^HARD ---

---

Figure 6-2: Semantic Representation - Initial
It can easily be seen how complicated the network can get, unless it is better organized. Also notice that I introduced a basic logic expression into the network. This was done for two reasons. First, to show how easy it can be to combine logic with other systems. Second, was to show how combining representations can make the representation simpler rather than more complicated. Normally, 3 separate nodes would be linked by 3 separate color links. The advantage of having separate nodes is that no logic procedure is needed, all inferences can be handled by the pattern matching scheme. The question marks indicate that more information is needed.
The frame representation of the knowledge shown previously would look something like this:

```
CARROT[                              VEGETABLE[
  ISA:VEGETABLE ]                     ISA:PLANT ]
GARDEN[
  GROW:PLANT ]
RABBIT[
  ISA:MAMMAL
  FOOD:CARROT
  COLOR:
  ;RANGE(BLACK v BROWN v WHITE) ]
FENCE[
  USETO:KEEPIN(?) v KEEPOUT(?) ]
HAMMER[
  USETO:HIT(?)
  PPROP:HEAVY ^ HARD ]
  PROP:HEAVY ^ HARD ]
```

Figure 6-3: Frame Representation - Initial

Again notice the use of logic in the representation and the use of question marks to indicate where information is needed. Also notice that ISA links are always the first line in the frame. The upper slots of a frame are usually fixed and the bottom slots are to be filled by specific instances of data. Although there is no absolute division of fixed and variable slots, I would think this would be a useful feature.
With this basic knowledge, the representation needs to represent the knowledge contained in the story itself. The representations given are the general representations. For example, the knowledge about all hammers is given in these representations. This representation is then copied and the copies are used to represent specific instances of the general object. So for example, when the story mentions the fact that John and Jane own a garden, this garden is given a specific name, such as GARDEN1. But GARDEN1 will inherit the knowledge known about gardens in general. If any knowledge becomes known about GARDEN1, it will be associated only with GARDEN1 and not with gardens in general.
The logic representation of the knowledge given in the story would look something like this:

\[
\begin{align*}
&\forall x. \text{GARDEN}^{-1}(x) \rightarrow \text{GARDEN}(x) \\
&\forall x. \text{VEGETABLE}^{-1}(x) \rightarrow \text{VEGETABLE}(x) \\
&\forall x. \text{RABBIT}^{-1}(x) \rightarrow \text{RABBIT}(x) \\
&\forall x. \text{JOHN}(x) \rightarrow \text{COLORBLIND}(x) \\
&\forall x. \text{FENCE}^{-1}(x) \rightarrow \text{FENCE} \\
&\forall x. \text{HAMMER}^{-1}(x) \rightarrow \text{HAMMER}(x) \\
&\text{OWNER}(x)
\end{align*}
\]

Figure 6-4: Logic Representation – Story

The function \text{OWNER} will return the value \text{JOHN} \^ \text{JANE} when \( x \) is \text{GARDEN}^{-1} and will return the value \text{JOHN} when \( x \) is \text{HAMMER}^{-1}. Functions serve the useful purpose of data acquisition. If the system is asked, \textit{who is the owner of the garden}, the system simply plugs in \text{GARDEN}^{-1} into the function \text{OWNER}, and the answer is neatly returned. But if the system is asked, \textit{who is colorblind}, it would have to locate the statement which represents this information. Because logic systems have no organization, this would be quite difficult with a large database.
The semantic representation of the knowledge contained in the story would look something like this:

```
COLORBLIND

---

| JOHN | own --> | GARDEN1 | own --> | JANE |

---

| HAMMER1 | PLANT |

---

| hit --> | ? |

---

| prop --> | HEAVY HARD |
```

Figure 6-5: Semantic Representation - Story

Notice how links are setup from the nodes HAMMER1 and GARDEN1, thus they inherit the information known about their entire group.
The frame representation of the knowledge contained in the story would look something like this:

\[
\begin{align*}
&\text{JOHN[} \\
&\text{OWN:GARDEN1} \\
&\text{VISION:COLORBLIND} \\
&\text{OWN:HAMMER1} \\
&\text{GARDEN1[} \\
&\text{GROW:VEGETABLE1} \\
&\text{VEGETABLE1[} \\
&\text{ISA:PLANT} \\
&\text{JANE[} \\
&\text{OWN:GARDEN1} \\
&\text{VISION:NORMAL} \\
&\text{HAMMER1[} \\
&\text{USETO:HIT(?)} \\
&\text{PROP:HEAVY & HARD} \\
\end{align*}
\]

Figure 6-6: Frame Representation - Story

Again notice that a complete copy of a frame is made, so each specific instance of a group inherits the knowledge known about all the members of that group. Also notice that Jane inherits the common trait of normal vision. Jane may not have normal vision, but unless stated otherwise, we assume that she does. Also note that with an actual large-scale system, the fact that Jane is a human, and assumed to be female, she may inherit hundreds of characteristics.
What inferences can be made from this knowledge?
The first inference, and the simplest, is that all carrots are plants. In the logic representation, the inference mechanism would use a theorem of the form:

\[ \text{IF } X=Y \text{ AND } Y=Z \text{ THEN } X=Z \]

With the semantic net and the frame representation, it would be a simple matter to follow the ISA links. Again the reasoning mechanism would need some kind of theorem to tell it the fact that:

\[ \text{IF } X \text{ ISA } Y \text{ AND } Y \text{ ISA } Z \text{ THEN } X \text{ ISA } Z \]

The next inference to be made is to find the color of RABBIT1. In logic the important expression is:

\[ \#X.\text{COLOR-RABBIT}(X) \]

When orange is used as an argument, the predicate COLOR-RABBIT will return the value NIL. When brown is used as an argument, the predicate will return the value TRUE. Thus you know that the color of RABBIT1 is brown. But the system must know how to use the knowledge, some sort of procedure or theorem would be needed.

With semantic nets, a network fragment would be created, and then an attempt would be made to match it.
The two fragments would look like this:

```
  ----------  ----------
  | RABBIT |---color---| ORANGE |
  ----------  ----------

  ----------  ----------
  | RABBIT |---color---| BROWN |
  ----------  ----------
```

Figure 6-7: Pattern Fragment - To match RABBIT

This would be a straightforward procedure if there was a separate link for each particular color. If there was only one link, a different procedure would be needed. The procedure would have to locate the color node, and then use a basic logic procedure to figure out the color.

A question often asked is why do we need to use logic at all? To answer this question, we need to look at a more complicated example. Let's assume that a RABBIT can be black, brown, or white, but NOT a combination of any of the 3 colors. With this type of relation, a different representation would be needed. As the OR and AND relations become more complex, a logic scheme would make the representation simpler.

In the frame representation it would be handled similarly to the logic representation. A logic procedure would be given the range and the possible values, it would then make the appropriate deductions.

Now if the representation were asked, what was the
fence for, how would it find the answer? First we have to consider the possibility that we need more information. The reasoning process would go something like this:

A fence is used to keep something in and to keep something out. A fence is to go around the garden, so it is to keep the contents of the garden in. The garden is used to grow plants, thus John wants to keep the plants in, in this would be VEGETABLE1. The RABBIT1 is also in the garden, but it doesn't belong there, so the fence is used to keep out the rabbit.

The system would know that GARDEN1 is inside the fence, hence the fence is used to keep in the GARDEN1. It would then assume that if the fence contains the garden, it also contains what is in the garden. Although this isn't exactly true, I will choose to ignore certain complexities. There is a small jump in the reasoning process, when it states that RABBIT1 doesn't belong in the garden. There must be more knowledge if any of the systems is to make the inference that the fence is to keep the rabbit out. Knowing the fact that a fence is use to keep something out isn't enough. There is a great deal of knowledge needed to make what is to us, a commonsense deduction. The necessary knowledge of what belongs in a garden and what doesn't is needed.
If the system is asked, could John use a rock instead of a hammer? The system would need to know how to compare objects, as described in Chapter 5. This is just one intelligent process the system would need if it was to act intelligently.

What if the system is asked, what type of vegetables are in the garden? Does the system have enough information? The reasoning process would go something like this:

The rabbit was eating the vegetables, since rabbits eat carrots, and we don't know of any other vegetables which rabbits eat, so we assume that the vegetables must be carrots.

In this particular case, it would seem that we have enough information. In logic, it would use a function called EQUAL. This function plugs in the two values given it into every possible predicate and function to see if they return identical results, along with checking all expressions. By giving the function CARROT and VEGETABLE1, it would find that all functions return identical values. For example, the function EAT would return RABBIT if X=CARROT and RABBIT1 if X=VEGETABLE1. The function would then use the expression:

\[ *x. \text{RABBIT1}(x) \rightarrow \text{RABBIT}(x) \]
to infer that the answers given are identical. Unfortunately, the system would have to check all functions, even those which are not applicable, again because of its lack of organization.

In semantic networks, a fragment would be created which would then be used to match fragments in the database. This fragment would match the RABBIT-EAT-CARROT fragment in the semantic network. The fragment is shown below.

```
| RABBIT | ---eat--> | ? |
```

Figure 6-8: Pattern Fragment - To match CARROT

For frame representations, the representation of a rabbit would look like:

```
RABBIT1[
  ISA: MAMMAL
  FOOD: CARROT
  COLOR: BROWN ]
```

Figure 6-9: RABBIT1 Frame - Version I

When the system would try to replace CARROT, it would instead note the fact that VEGETABLE1 is a carrot. This would of course be in the frame representing VEGETABLE1.
If the system were told that the rabbit was eating lettuce, you would probably want to include this fact in the rabbit frame. The frame would then look like this:

```
RABBIT1[
  ISA:MAMMAL
  FOOD:CARROT ^ LETTUCE
  COLOR:BROWN ]
```

**Figure 6-10:** RABBIT1 Frame - Version II

Note that VEGETABLE1 and LETTUCE are handled differently, this is because VEGETABLE1 is a variable, and LETTUCE is not.

I think it has been shown that the frame organization is better suited for language comprehension. By combining basic logic, it can be good for problem solving also.
7. Representations for Problem Solving and Language Comprehension

7.1 Description

Most representations fall into the category of being either good for problem solving or good for language comprehension. Is there one representation that can do both, should there be one representation for both? Rieger [11] and Charniak [12] have written papers on possible representations for both predicate calculus and language comprehension. Their representations are purely hypothetical, and have not been implemented and tested. Rieger discusses the representation of dynamic knowledge from an abstract point of view. Charniak discusses both static and dynamic knowledge from a implementation point of view.

Charniak suggests combining aspects of both frame and predicate calculus systems together to get the benefits of both. Frames offer a natural way to partition knowledge and reduce the domain of knowledge. Predicate calculus enables data to be stored in small chunks and allows deep inferences. Each predicate calculus statement is associated with a frame so a statement can not be used unless its frame is active.
Requirements of problem solving are satisfied by using predicate calculus as a base language, and therefore it has a predicate calculus deductive component.

Charniak attempts to make predicate calculus format seem more natural by introducing abbreviations. I personally found the abbreviations and conventions confusing. He also was constantly changing notation and syntax, as he added to the representation. Because the system is basically a predicate calculus system, the problems of predicate calculus are carried over. The main problem with predicate calculus is that there is no way of representing ambiguity or imprecise knowledge.

Rieger proposes a new theory to represent commonsense algorithmic world knowledge combined with large-scale organization of the memory based on bypassable causal selection networks. Commonsense algorithmic world knowledge is what Rieger calls dynamic knowledge, which also includes performance knowledge. By using a set of 26 different links, he proposes that all dynamic knowledge can be represented. Rieger incorporates information into the links themselves. This knowledge is arranged by using bypassable causal selection networks, given a goal state or goal statechange, the network selects the most relevant algorithm pattern for achieving that goal state or
statechange. Bypassable causal selection networks are similar to the normal networks, in which tests are made at each node to determine what branch to take, except that once a particular path is followed in a certain environment, a bypass is setup for that particular environment. Therefore when the same environment or conditions occur, the bypass is followed, thereby increasing efficiency. This adaptation is actually learning. At the terminal nodes of these networks are what are called approaches, expressed in the commonsense algorithm patterns.
7.2 Discussion

Now let's illustrate the strengths and weaknesses of each respective system with some examples. Given the sentence, Cloudy skies may mean it's going to rain. In Rieger's system, we would have a want link between cloudy skies and rain indicating that if there are cloudy skies, then it might rain. However, the opposite is not true. Both systems could easily represent the fact If it rains, then the skies must be cloudy. Rieger's system is really a variation of semantic networks, and has many of the same problems. One such problem is the combinatorial problem of a large database. Rieger suggests that this won't be that great of a problem because the network will not be that deep, just wide. Also, because of the bypasses, frequently used knowledge or commonsense knowledge can be quickly accessed.

Now let's consider a robot that is programmed to recharge itself. This robot would have to find electrical outlets. Assuming it has the ability to see, it would try to find outlets much as a human would. With the knowledge that outlets are usually located in walls about a foot off the floor, the robot would search each wall, in each room, till it found an outlet. Assume that there is only one outlet in a particular building. Once this outlet has been found, in Rieger's system, a bypass
is planted, so whenever the robot is in the same building it will know where the outlet is and will not have to search the building. Charniak's system does not have this capability.

Rieger's system can be broken down into three parts. The first part is static knowledge system, which I suspect would use the frame system. The second part is the dynamic knowledge system which utilizes the 26 different links. The third part is the network which organizes the knowledge. Charniak's system consists of the first and third parts. Charniak's paper was really not to propose a solution but to simply push the idea that it is possible to find a representation system that would satisfy both the needs of language comprehension and problem solving. The actual formalism presented was secondary.

Anything that can be represented in Charniak's system can be represented in Rieger's system. Suppose you wanted a robot to be able to use the water fountain. With Charniak's system, there would be a frame for the water fountain, in this frame there would be a slot designated for the operation of the fountain. This slot would contain the primitive action, PUSH. However, for the water fountain, you must push constantly; for a radio, you would only have to push once to turn it on. A
separate primitive action would be needed for what is called a one shot push. Because Rieger’s uses different links, he would have a primitive action push and then link it using a continuous link for the water fountain and a one-shot link for the radio.

Another problem is the use of vague terms, such as "little". For example, how much time is "a little while" or how much is "a little more". Each person has his/her own ideas of how much is a little. In both systems, a predetermined number would have to be associated with such vague terms. One problem common to both systems is the inability to represent time relations between actions and events.

Rieger views language comprehension as that process which elucidates the interrelationships among a collection or sequence of thoughts by consulting the kinds of world knowledge stored in an algorithmic memory. The basic character of the language comprehension process is one of prediction and fulfillment, wherein every perception gives rise to general expectations about what might follow. Rieger illustrates this by an example which contains actors performing actions, and then predicting what actions will come next. While this is fine for children’s stories, language comprehension involves a lot more than predicting actions and
fulfilling goals.

Rieger never mentions how he would represent time relations, where questions, and the representation of words such as "should" and "could". These are the most common problems in today's language comprehension systems. One advantage that his system would have is the ability to predict a particular actor's actions quickly based on prior experience. RVnasses would be setup for certain actors in certain situations. When these situations arise, a prediction of the actor's actions can be quickly made.

It is important to remember that most language comprehension involves the transfer of information or knowledge. The system must have the ability to interpret and store this information effectively, a subject that neither representation discussed.
8. Conclusions

The organization of representations has become more complex. The first representation was logic, with very little organization, then semantic networks, and now frames. There seems to be a trend toward the more elaborate representations. The more complex organizations will make application of the intelligent processes easier. The ability to store information in small chunks will facilitate the problem solving process. The ability to group this information will facilitate the language comprehension process. I envision the representation of the future to be well structured, with many levels of organization. The lower levels will be more detailed than the upper levels. Each level will be individually accessible, so different intelligent processes can use different levels which are best suited for their particular processes. I would also suspect that certain concepts such as time relations, will be handled by separate representation systems.

I used two terms throughout this paper; representation and system. When I use the term representation, I meant the organizational structure itself. When I use the term system, I meant the organizational structure, the intelligent programs, the cross-reference dictionaries, and everything else which
is needed for the entire system to function. The representation is the base for the system. If it is a good one, the rest of the system should fall in place.
References


Vita

Thomas John Evans III

Born May 31, 1960 in Richmond, Virginia, the son of Thomas John Evans Jr. and Nancy Jane Evans. Attended Widener University from September 1978 to May 1982. Graduated Magna Cum Laude with the degree of Bachelor of Science in Engineering.