be this representation of the fact "In California, Grass and Trees produce green regions."

```
(To-Establish (GreenRegion x)
  Establish (AND (InCalifornia())
    (OR (Establish (Grass x))
    (Establish (Trees x)))))
```

This might mean: To infer that *x* is a green region, establish that you are in California and then try to establish that *x* arose from grass. Should the grass inference fail, try to establish that *x* arose from trees. Since the full power of the programming language is available to an Establish statement, it can perform general computations to establish the inference.

The important point here: Rather than a set of clauses whose application must be organized by an interpreter, propositions may be represented by an explicit control sequence, including procedure calls to other programs. In the example, (Grass *x*) and (Trees *x*) may be procedures which have their own complicated control structures.

To say that in a computer "everything is propositions" is a truism; any program can be reduced to a Turing machine described by a finite set of "propositions" with a very simple rule of "inference." The issue is at what level the program should be described. A program may be doing propositional resolution theorem proving or analogical trajectory planning with three-dimensional models; it is not helpful to blur this basic functional distinction by appealing to the lowest implementational level.

10.2 SEMANTIC NETS

10.2.1 Semantic Net Basics

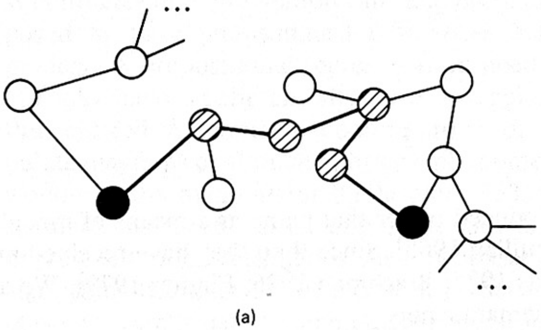
Semantic nets were first introduced under that name as a means of modeling human associative memory [Quillian 1968]. Since then they have received much attention [Nilsson 1980; Woods 1975; Brachman 1976; Findler 1979]. We are concerned with three aspects of semantic nets.

1. Semantic nets can be used as a data structure for conveniently accessing both analogical and propositional representations. For the latter their construction is straightforward and based solely on propositional syntax (Chapter 12).
2. Semantic nets can be used as an analogical structure that mirrors the relevant relations between world entities.
3. Semantic nets can be used as a propositional representation with special rules of inference. Both classical and extended inference can be supported, but it is a challenging enterprise to design net structure that provides the properties of formal logic [Schubert 1976; Hendrix 1979].

A semantic network represents objects and relationships between objects as a graph structure of *nodes* and (labeled) *arcs*. The arcs usually represent relations between nodes and may be "followed" to proceed from node to node. A directed arc with label L between nodes X and Y can signify that the predicate $L(X, Y)$ is true. If, in addition, it has a value V, the arc can signify that some function or relation holds: $L(X, Y) = V$.

The *indexing property* of a network is one of its useful aspects. The network can be constructed so that objects that are often associated in computations, or are especially relevant or conceptually close to each other, may be represented by nodes in the network that are near each other in the network (as measured by number of arcs separating them). Figure 10.3 shows these ideas: (a) nodes can be associated by searching outward along arcs and (b) nodes near a specified node are readily available by following arcs. Semantic networks are especially attractive as analogical representations of spatial states of affairs. If we restrict ourselves to binary spatial relations ("above," and "west of," for example), physical objects or parts of objects may be represented by nodes, and their positions with respect to each other by arcs.

Let us look at a semantic net and make some basic observations. Figure 10.4 is meant to be an analogical representation of an arrangement of chairs around a table. The LEFT-OF and RIGHT-OF relations are directed arcs, the ADJACENT relation is undirected; there can be several such undirected arcs between nodes. Note here that the LEFT-OF and RIGHT-OF relations do not behave in their normal way. If they are transitive, as is normal, then every chair is both LEFT-OF and



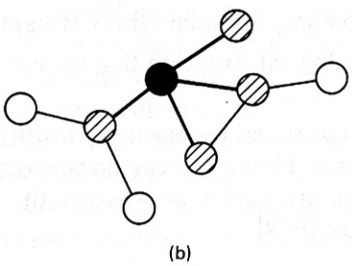


Fig. 10.3 Semantic networks as structures for associative search. (a) Associating two nodes. (b) Retrieving nearby nodes.

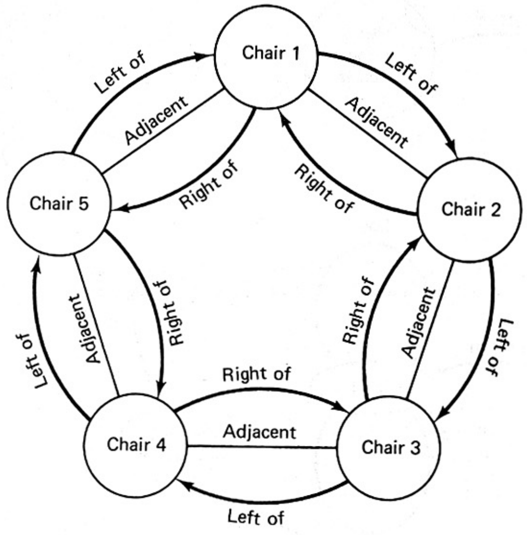


Fig. 10.4 A representation of chairs at a table.

RIGHT-OF every other chair. Flexible treatment of this sort of phenomenon is sometimes difficult in propositional representations.

A simple but basic point: The net of Fig. 10.4 seems to say interesting things about furniture in a scene. But notice that merely by rewriting labels the same net could be "about" modular arithmetic, a string of pearls, or any number of things. There are two morals here. First, a sparsely connected representation (analogical or propositional) may have several equally good interpretations. Second, a net without any interpretation procedures essentially represents nothing [McDermott 1976].

Now consider three neighboring chairs described by the following relations.

1. CHAIR(Armchair), CHAIR(Highchair), CHAIR(Stool)
2. WIDE(Armchair)
3. HIGH(Highchair)
4. LOW(Stool)
5. LEFT-OF(Armchair, Highchair)
6. LEFT-OF(Highchair, Stool)
7. BETWEEN(Highchair, Armchair, Stool)

The relations include four properties (relations with "one argument"), a two-argument and a three-argument relation. One way to encode this information in a net is shown in Fig. 10.5a. Nodes represent individuals, and properties are kept as node contents. The directed arcs represent only binary relations, and "betweenness" is left implicit. Properties can equally well be represented as labeled arcs (Fig. 10.5b).

Relations are encoded as nodes in Fig. 10.6. Here the BETWEEN relation is encoded asymmetrically: it is not possible to tell by arcs emanating from the stool that it is in a "between" relationship.

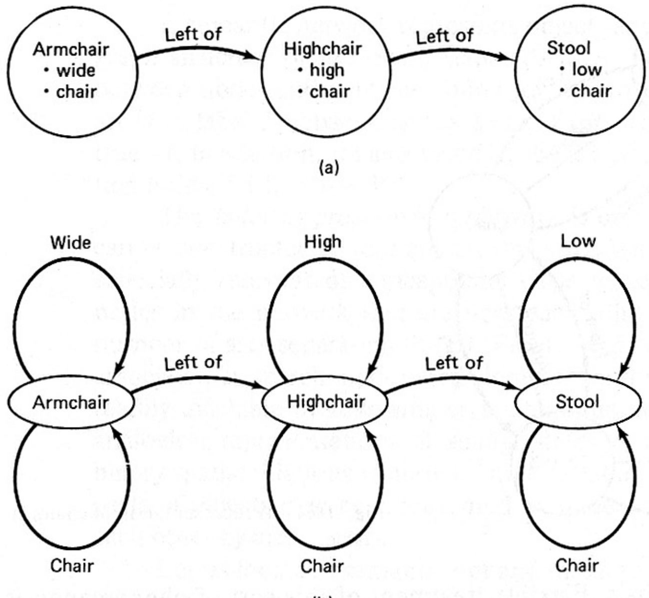


Fig. 10.5 (a) A simple semantic net. (b) An equivalent net.

The three-place relation is treated more symmetrically in Fig. 10.7. In general, *n*-place relations may be "binarized" this way; create a node for the "relation instance" and new (relation) nodes for each distinct argument role in the *n*-ary relation.

An important point: Arcs and nodes had a uniform semantics in Fig. 10.4. This property was lost in the succeeding nets; nodes are either "things" or relations, and arcs leading into relations are not the same as those leading out. For such nets to be useful, the net interpreter (a program that manipulates the net) must keep these things straight. It is possible but not easy to devise a rich and uniform network semantics [Brachman 1979].

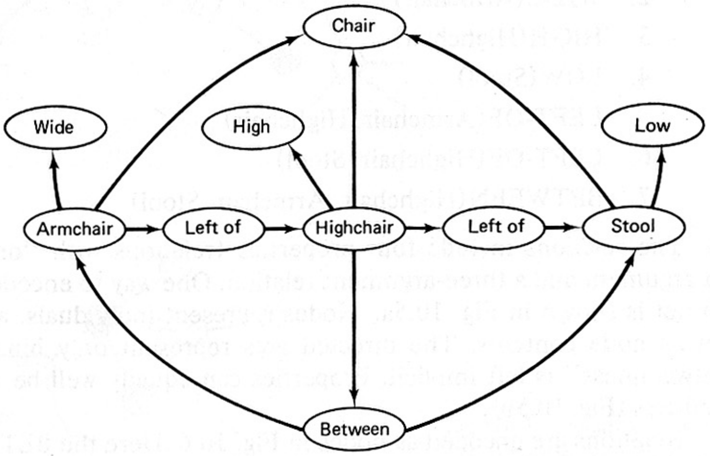


Fig. 10.6 A net with more explicit information.

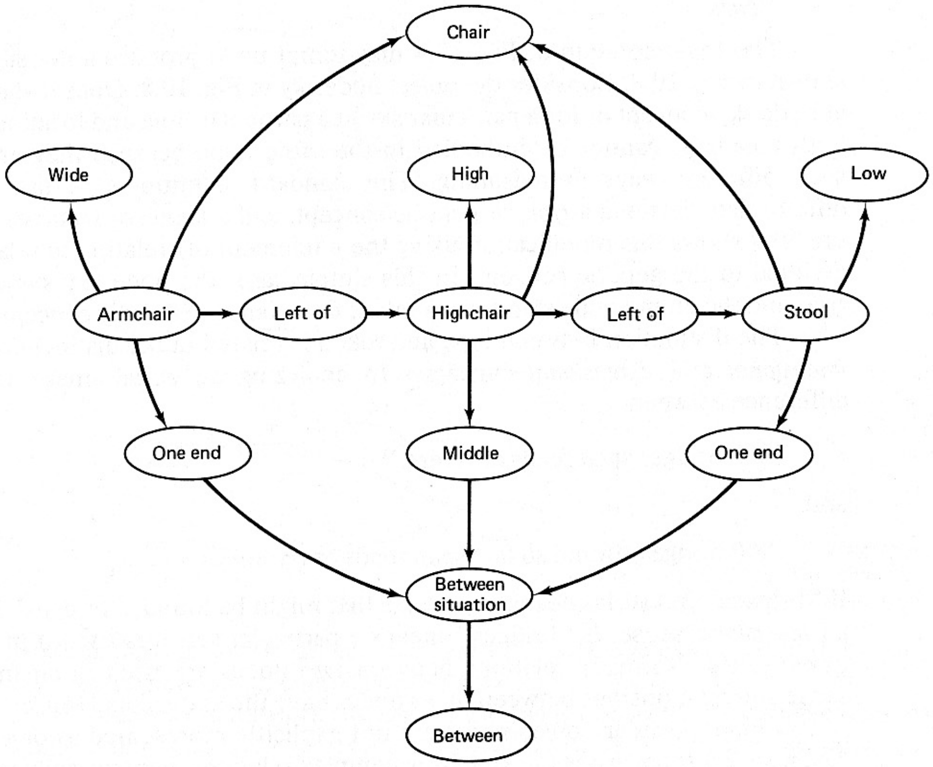


Fig. 10.7 A net with yet more explicit information.

10.2.2 Semantic Nets for Inference

This section explores some further important issues in the semantics of semantic nets. In Chapter 12 semantic nets are used as an indexing mechanism in predicate calculus theorem proving. In some applications an inference system with provably good formal properties may be too restrictive. Some formal properties (such as maintaining consistency by not deducing contradictions) may be considered vital, however. How can "good behavior" be obtained from a representation that may contain "inconsistent" information?

One example of an "inconsistent" representation is the net of Fig. 10.3, with its LEFT-OF and RIGHT-OF problem. Another example is a net version of the propositions "All birds fly," "Penguins are birds," "Penguins do not fly." The generalization is useful "commonsense" knowledge, but the rare exceptions may be important, too. Network interpreters can cope with these sorts of problems by a number of methods, such as only accessing a consistent subnetwork, making deductions from the particular toward the general (this takes care of penguins), and so forth. All these techniques depend on the structure imposed by the net.

Some more subtle aspects of net representations appear below.

Nodes

The basic notation of Fig. 10.4 may tempt us to produce a net such as that shown in Fig. 10.8. Consider the object node sky in Fig. 10.8. Does it stand for the generic sky concept or for a particular sky at a particular time and location? Clearly both meanings cannot be embodied in the same node because they are used in such different ways in reasoning. The standard solution is to use nodes to differentiate between a *type*, or generic concept, and a *token*, or instance of it. Figure 10.9 shows this modification using the *e* (element of) relation to relate the individual to the generic concept. In this simple case, the node sky stands for the *type*, and the empty node stands for a *token*, or instance of the sky concept.

The distinction between type and token is related to the distinction between *intensional* and *extensional* concepts. In analyzing an aerial image there is a difference between

"All bridges span roads or rivers." (10.1)

and

"All bridges (found so far) span roads or rivers." (10.2)

If "bridges" in (10.1) means *any* bridge that might be found, "bridges" is used in an *intensional* sense. If "bridges" means a particular set, it is used it in an *extensional* sense. Normally relations between *type* nodes are used in an intensional sense and relationships between *token* nodes have the extensional sense.

Virtual nodes are objects that are not explicitly represented as object nodes. The need for them arises in expressing complex relations. For example, consider

"The bridge that is at the intersection of road 57
and river 3 is near building 30." (10.3)

which may be represented as shown in Fig. 10.10. The node labeled *x* is the result of intersecting a particular road with a particular river. It is not represented explicitly as an instance of any generic concept; it is a *virtual node.* Virtual nodes can be eliminated by introducing very complex relations, but this would sacrifice an important property of networks, the ability to build up a very large number of com-

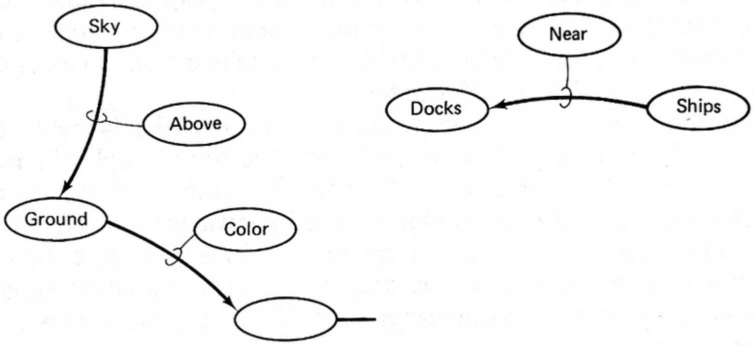
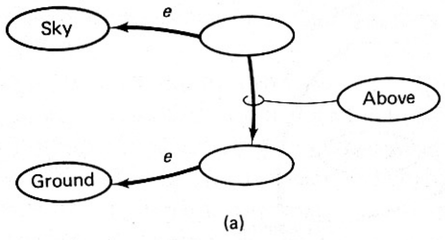


Fig. 10.8 Type or token nodes?



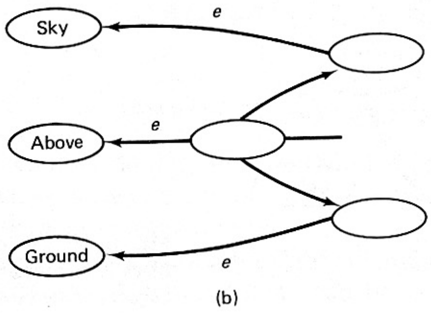


Fig. 10.9 Distinguishing between types and tokens: (a) Tokenizing an instance. (b) Tokenizing an assertion.

plex relations from a small set of primitives. Virtual nodes enhance this ability by referring to portions of complicated relations.

Nodes in the network can also be used as *variables.* These variables can match other nodes which represent constants. In Fig. 10.11, x and y are variables and the rest of the nodes are constants. If node x is matched to the "telephone" node, then x can be regarded as a "telephone" node.

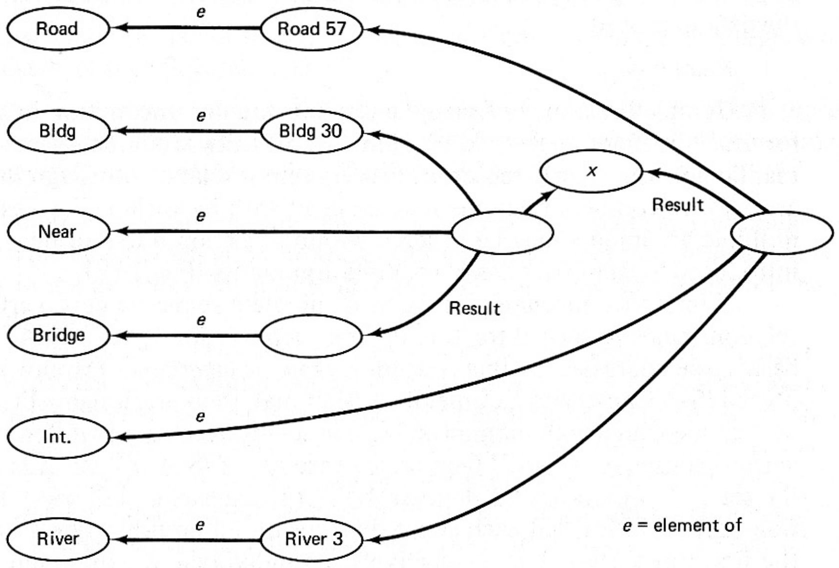
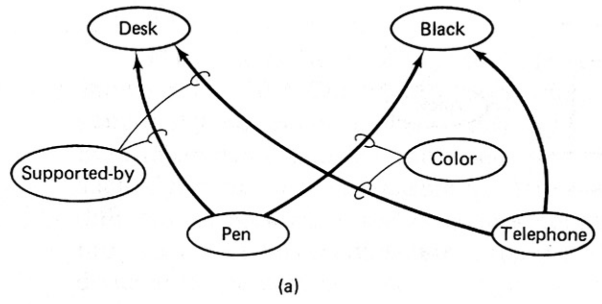


Fig. 10.10 Virtual nodes.



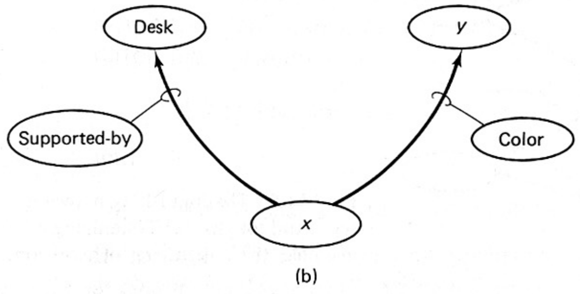


Fig. 10.11 Nodes as variables. (a) Black telephone and pen on desk. (b) Object denoted by variable x with variable color y.

Often, it is useful to have numerical values as node properties. This can extend the discrete representation of nodes and arcs to a continuous one. For example, in addition to "color of x is red37" we may also associate the particular value of red that we mean with node red37. A special kind of value is a *default value*. If a value can be found for the node in the course of matching other nodes with values or by examining image data, then that value is used for the node value. Otherwise, the default is used.

Relations

Complex relations of many arguments are not uncommon in the world, but for the bulk of practical work, relations of only a few arguments seem to suffice. Semantic nets can clearly represent two-argument relations through their nodes and arcs. More complex relations may be dealt with by various devices. The links to multiple arguments may be ordered within a relation node, or new nodes may be introduced to label the roles of multiple arguments (Fig. 10.7).

If inference mechanisms are to manipulate semantic nets, certain important relations deserve special treatment. One such relation is the "IS-A" relation. The basic issue addressed by this relation is *property inheritance* [Moore 1979]. That is, if Fred IS-A Camel and a Camel IS-A Mammal, then presumably Fred has the properties associated with mammals. It often seems necessary to differentiate between various senses of "IS-A." One basic sense of "X IS-A Y" is "X is an element of the set Y"; others are "X denotes Y," "X is a subset of Y," and "Y is an abstraction of X." Notice that each sense depends on differently "typed" arguments; in the first three cases X is, respectively, an individual, a name, and a set. Deeper treatments of these issues are readily available [Brachman 1979; Hayes* 1977; Nilsson 1980].

It is particularly helpful to have a denotion link to keep perceptual structures separate from model structures. Then if mistakes are made by the vision automaton, a correction mechanism can either sever the denotation link completely or create a new denotation link between the correct model and image structures.

When dealing with many spatial relations, it is economical to recognize that many relations are "inverses" of each other. That is, LEFT-OF(x,y) is the "inverse" of RIGHT-OF(x,y);

$$\text{LEFT-OF}(x,y) \iff \text{RIGHT-OF}(y,x)$$

and also

$$\text{ADJACENT}(x,y) \iff \text{ADJACENT}(y,x)$$

Rather than double the number of these kinds of links, one can *normalize* them. That is, only one half of the inverse pair is used, and the interpreter infers the inverse relation when necessary.

Properties have a different semantics depending on the type of object that has the property. An "abstract" node can have a property that gives one aspect or refinement of the represented concept. A property of a "concrete" node presumably means an established and quantified property of the individual.

Partitions

Partitions are a powerful notion in networks. "Partition" is not used in the sense of a mathematical partition, but in the sense of a barrier. Since the network is a graph, it contains no intrinsic method of delimiting subgraphs of nodes and arcs. Such subgraphs are useful for two reasons:

1. *Syntactic.* It is useful to delimit that part of the network which represents the results of specific inferences.
2. *Semantic.* It is useful to delimit that part of the network which represents knowledge about specific objects. Partitions may then be used to impose a hierarchy upon an otherwise "flat" structure of nodes.

The simple way of representing partitions in a net is to create an additional node to represent the partition and introduce additional arcs from that node to every node or arc in the partition. Partitions allow the nodes and relations in them to be manipulated as a unit.

Notationally, it is cleaner to draw a labeled boundary enclosing the relevant nodes (or arcs). An example is shown by Fig. 10.12 where we consider two objects each made up of several parts with one object entirely left of the other. Rather than use a separate LEFT-OF relation for each of the parts, a single relation can be used between the two partitions. Any pair of parts (one from each object) should inherit the LEFT-OF relation. Partitions may be used to implement quantification in semantic net representations of predicate calculus [Hendrix 1975, 1979]. They may be used to implement frames (Section 10.3.1).

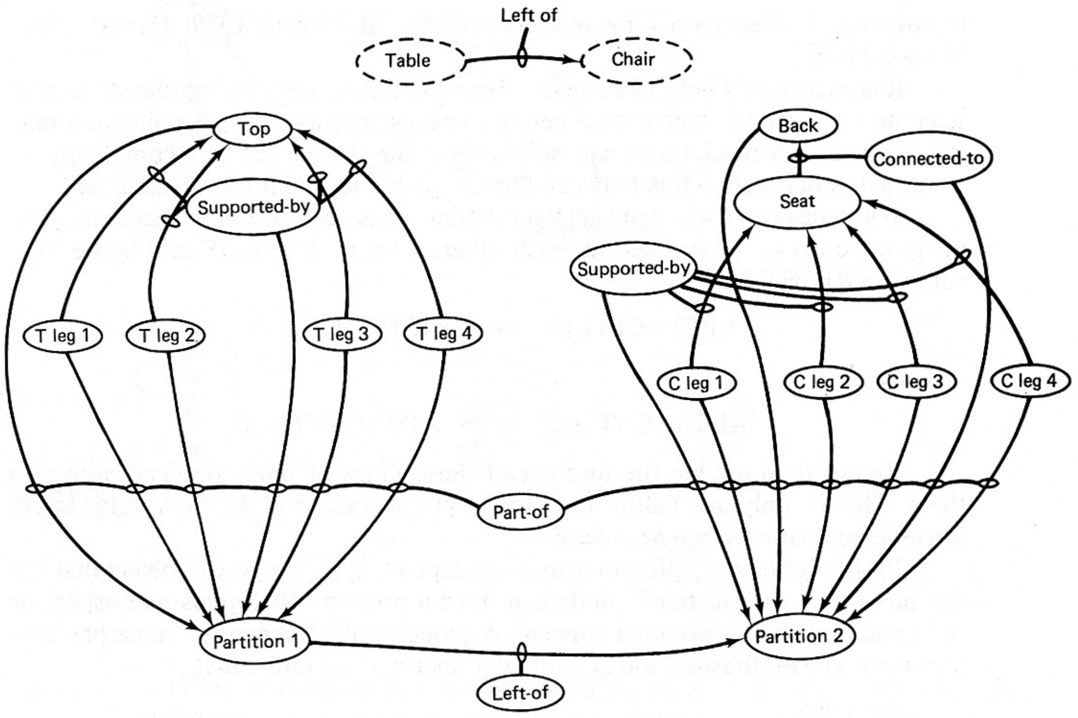


Fig. 10.12 The use of partitions. (a) Construction of a partition. (b) Two objects described by partitions.

Conversions

It is important to be able to transform from geometric (and logical) representations to propositional abstract representations and vice versa. For example, in Fig. 10.13 the problem is to find the exact location of a telephone on a previously located desk. In this case, propositional knowledge that telephones are usually on desktops, together with the desk top location and knowledge about the size of telephones, define a search area in the image.

Converting image data about a particular group of objects into relational form involves the inverse problem. The problem is to perform a level of abstraction to remove the specificity of the geometric knowledge and derive a relation that is appropriate in a larger context. For example, the following program fragment creates the relations ABOVE(A,B), where A and B are world objects.

Comment: assume a world coordinate system where Z is the positive vertical.

```
    Find $ZA_{\min}$ for $Z$ in $A$ and $ZB_{\max}$ for $Z$ in $B$.
    If $ZA_{\min} > ZB_{\max}$, then make ABOVE $(A,B)$ true.
```

Many other definitions of ABOVE, one of which compares centers of gravity, are possible. In most cases, the conversion from continuous geometric relations to discrete propositional relations involves more or less arbitrary conventions. To appreciate this further, consult Fig. 10.14 and try to determine in which of the cases

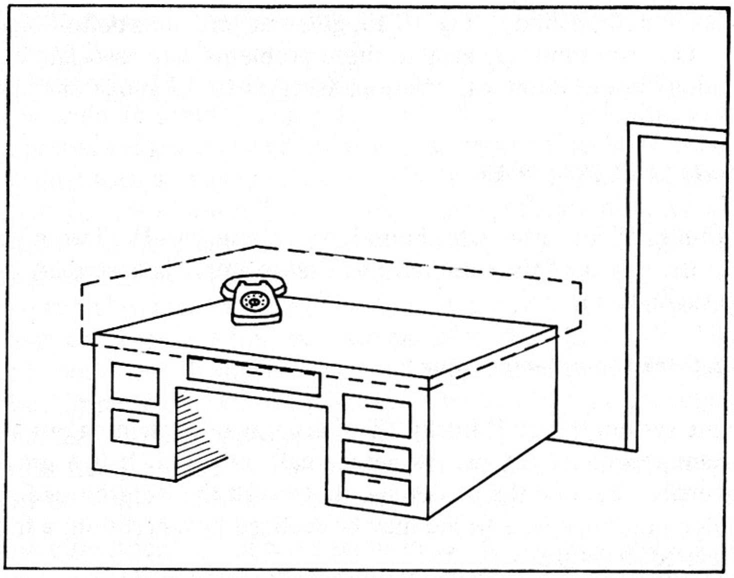


Fig. 10.13 Search area defined by relational bindings.

block *A* is LEFT-OF block *B*. Figure 10.14d shows a case where different answers are obtained depending on whether a two-dimensional or three-dimensional interpretation is used. Also, when relations are used to encode what is *usually* true of the world, it is often easy to construct a counterexample. Winston [Winston 1975] used

SUPPORTS (B,A) ABOVE (A,B)

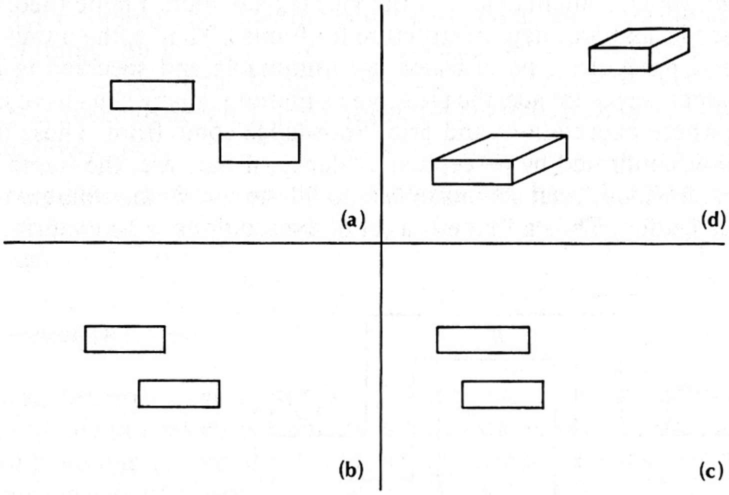


Fig. 10.14 Examples to demonstrate difficulties in encoding spatial relation LEFT-OF (see text).

which is contradicted by Fig. 10.15, given the previous definition of ABOVE.

One common way around these problems is to associate quantitative, "continuous" information with relations (section 10.3.2 and later examples).

10.3 SEMANTIC NET EXAMPLES

Examples of semantic nets abound throughout Part IV. Two more examples illustrate the power of the notions. The first example is described very generally, the second in detail.

10.3.1 Frame Implementations

Frame system theory [Minsky 1975] is a way of explaining our quick access to important aspects of a (perhaps perceptual) situation. It is a provocative and controversial idea, and the reader should consult the References for a full treatment. Implementationally, a frame may be realized by a partition; a frame is a "chunk" of related structure.

Associating related "chunks" of knowledge into manipulable units is a powerful and widespread idea in artificial intelligence [Hayes 1980; Hendrix 1979] as well as psychology. These chunks go by several names: units, frames, partitions, schemata, depictions, scripts, and so forth [Schank and Abelson 1977; Moore and Newell 1973; Roberts and Goldstein 1977; Hayes* 1977; Bobrow and Winograd 1977, 1979; Stefik 1979; Lehnert and Wilks 1979; Rumelhart et al. 1972].

Frames systems incorporate a theory of associative recall in which one selects frames from memory that are relevant to the situation in which one finds oneself. These frames include several kinds of information. Most important, frames have *slots* which contain details of the viewing situation. Frame theory dictates a strictly specific and prototypical structure for frames. That is, the number and type of slots for a particular type of frame are immutable and specified in advance. Further, frames represent specific prototype situations; many slots have default values; this is where expectations and prior knowledge come from. These default values may be disconfirmed by perceptual evidence; if they are, the frame can contain information about what actions to take to fill the slot. Some slots are to be filled in by investigation. Thus a frame is a set of expectations to be confirmed or disconfirmed

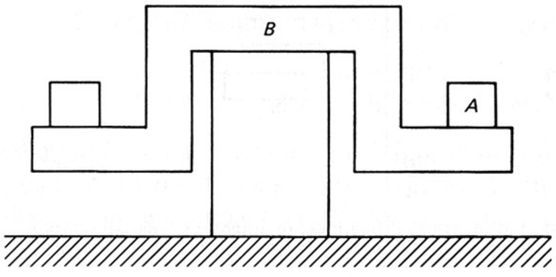


Fig. 10.15 A counterexample to SUPPORTS$(B, A) \Rightarrow$ ABOVE(A, B).