14 REPRESENTING MEANING

ISHMAEL: *Surely all this is not without meaning.*
Herman Melville, *Moby Dick*

The approach to semantics that is introduced here, and is elaborated on in the next four chapters, is based on the notion that the meaning of linguistic utterances can be captured in formal structures, which we will call **meaning representations**. Correspondingly, the frameworks that are used to specify the syntax and semantics of these representations will be called **meaning representation languages**. These meaning representations play a role analogous to that of the phonological, morphological, and syntactic representations introduced in earlier chapters.

MEANING REPRESENTATIONS

MEANING REPRESENTATION LANGUAGES

The need for these representations arises when neither the raw linguistic inputs, nor any of the structures derivable from them by any of the transducers we have studied, facilitate the kind of semantic processing that is desired. More specifically, what is needed are representations that can bridge the gap from linguistic inputs to the kind of non-linguistic knowledge needed to perform a variety of tasks involving the meaning of linguistic inputs.

To illustrate this idea, consider the following everyday language tasks that require some form of semantic processing:

- answering an essay question on an exam
- deciding what to order at a restaurant by reading a menu
- learning to use a new piece of software by reading the manual
- realizing that you've been insulted
- following a recipe

It should be clear that simply having access to the kind of phonological, morphological, and syntactic representations we have discussed thus far will not

get us very far on accomplishing any of these tasks. These tasks require access to representations that link the linguistic elements involved in the task to the non-linguistic *knowledge of the world* needed to successfully accomplish them. For example, some of the knowledge of the world needed to perform the above tasks includes:

- Answering and grading essay questions requires background knowledge about the topic of the question, the desired knowledge level of the students, and how such questions are *normally* answered.
- Reading a menu and deciding what to order, giving advice about where to go to dinner, following a recipe, and generating new recipes all require deep knowledge about food, its preparation, what people like to eat and what restaurants are like.
- Learning to use a piece of software by reading a manual, or giving advice about how to do the same, requires deep knowledge about current computers, the specific software in question, similar software applications, and knowledge about users in general.

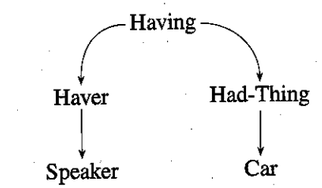
In the representational approach being explored here, we take linguistic inputs and construct meaning representations that are made up of the *same kind of stuff* that is used to represent this kind of everyday commonsense knowledge of the world. The process whereby such representations are created and assigned to linguistic inputs is called **semantic analysis**.

SEMANTIC ANALYSIS

To make this notion more concrete, consider Figure 14.1, which shows sample meaning representations for the sentence *I have a car* using four frequently used meaning representation languages. The first row illustrates a sentence in **First Order Predicate Calculus**, which will be covered in detail in Section 14.3; the graph in the center illustrates a **Semantic Network**, which will be discussed further in Section 14.5; the third row contains a **Conceptual Dependency** diagram, discussed in more detail in Chapter 16, and finally a frame-based representation, also covered in Section 14.5.

While there are a number of significant differences among these four approaches to representation, at an abstract level they all share as a common foundation the notion that a meaning representation consists of structures composed from a set of symbols. When appropriately arranged, these symbol structures are taken to correspond to objects, and relations among objects, in some world being represented. In this case, all four representations make use of symbols corresponding to the speaker, a car, and a number of relations denoting the possession of one by the other.

$\exists x,y Having(x) \wedge Haver(Speaker,x) \wedge HadThing(y,x) \wedge Car(y)$



Car
⇑ POSS-BY
Speaker

```
Having
 Haver:   Speaker
 HadThing:  Car
```

Figure 14.1 A list of symbols, two directed graphs, and a record structure: a sampler of meaning representations for *I have a car.*

It is important to note that these representations can be viewed from at least two distinct perspectives in all four of these approaches: as representations of the meaning of the particular linguistic input *I have a car*, and as representations of the state of affairs in some world. It is this dual perspective that allows these representations to be used to link linguistic inputs to the world and to our knowledge of it.

The structure of this part of the book parallels that of the previous parts. We will alternate discussions of the nature of meaning representations with discussions of the computational processes that can produce them. More specifically, this chapter introduces the basics of what is needed in a meaning representation, while Chapter 15 introduces a number of techniques for assigning meanings to linguistic inputs. Chapter 16 explores a range of complex representational issues related to the meanings of words. Chapter 17 then explores some robust computational methods designed to exploit these lexical representations.

Note that since the emphasis of this chapter is on the basic requirements of meaning representations, we will defer a number of extremely important issues to later chapters. In particular, the focus of this chapter is on representing what is sometimes called the **literal meaning** of sentences.

LITERAL MEANING

By this, we have in mind representations that are closely tied to the con-
ventional meanings of the words that are used to create them, and that do
not reflect the context in which they occur. The shortcomings of such repre-
sentations with respect to phenomena such as idioms and metaphor will be
discussed in the next two chapters, while the role of context in ascertaining
the deeper meaning of sentences will be covered in Chapters 18 and 19.

There are three major parts to this chapter. Section 14.1 explores some
of the practical computational requirements for what is needed in a meaning
representation language. Section 14.2 then discusses some of the ways that
language is structured to convey meaning. Section 14.3 then provides an
introduction to First Order Predicate Calculus, which has historically been
the principal technique used to investigate semantic issues.

14.1 Computational Desiderata for Representations

We begin by considering the issue of why meaning representations are needed
and what they should do for us. To focus this discussion, we will consider in
more detail the task of giving advice about restaurants to tourists. In this dis-
cussion, we will assume that we have a computer system that accepts spoken
language queries from tourists and construct appropriate responses by using
a knowledge base of relevant domain knowledge. A series of examples will
serve to introduce some of the basic requirements that a meaning represen-
tation must fulfill, and some of the complications that inevitably arise in the
process of designing such meaning representations. In each of these exam-
ples, we will examine the role that the representation of the meaning of the
request must play in the process of satisfying it.

Verifiability

Let us begin by considering the following simple question:

(14.1) Does Maharani serve vegetarian food?

This example illustrates the most basic requirement for a meaning represen-
tation: it must be possible to use the representation to determine the relation-
ship between the meaning of a sentence and the world as we know it. In other
words, we need to be able to determine the truth of our representations. The
most straightforward way to implement this notion is make it possible for a
system to compare, or *match*, the representation of the meaning of an input
against the representations in its **knowledge base**, its store of information

KNOWLEDGE BASE

its world.

In this example, let us assume that the meaning of this question con-
as a component, the meaning underlying the proposition *Maharani*
serves vegetarian food. For now, we will simply gloss this representation as:

Serves(*Maharani*, *VegetarianFood*)

It is this representation of the input that will be matched against the
owledge base of facts about a set of restaurants. If the system finds a
epresentation matching the input proposition in its knowledge base, it can
turn an affirmative answer. Otherwise, it must either say *No*, if its knowl-
ge of local restaurants is complete, or say that it does not know if there is
ason to believe that its knowledge is incomplete.

This notion is known as **verifiability**, and concerns a system's ability VERIFIABILITY
o compare the state of affairs described by a representation to the state of
ffairs in some world as modeled in a knowledge base.[1]

Unambiguous Representations

The domain of semantics, like all the other domains we have studied, is
subject to ambiguity. Specifically, single linguistic inputs can legitimately
have different meaning representations assigned to them based on the cir-
cumstances in which they occur.

Consider the following example from the BERP corpus:

(14.2) I wanna eat someplace that's close to ICSI.

Given the allowable argument structures for the verb *eat*, this sentence can
either mean that the speaker wants to eat *at* some nearby location, or under
a Godzilla as speaker interpretation, the speaker may want to devour some
nearby location. The answer generated by the system for this request will
depend on which interpretation is chosen as the correct one.

Since ambiguities such as this abound in all genres of all languages,
some means of determining that certain interpretations are preferable (or
alternatively less preferable) than others is needed. The various linguistic
phenomenon that give rise to such ambiguities, and the techniques that can
be employed to deal with them, will be discussed in detail in the next four
chapters.

Our concern in this chapter, however, is with the status of our meaning
representations with respect to ambiguity, and not with how we arrive at

[1] This is a fairly practical characterization of verifiability. More theoretical views of this notion are briefly covered in Section 14.6.

correct interpretations. Since we reason about, and act upon, the seman content of linguistic inputs, the final representation of an input's meanin should be free from any ambiguity. Therefore, regardless of any ambiguity in the raw input, it is critical that a meaning representation language supp representations that have a single unambiguous interpretation.[2]

VAGUENESS

A concept closely related to ambiguity is **vagueness**. Like ambiguity, vagueness can make it difficult to determine what to do with a particular input based on its meaning representation. Vagueness, however, does not give rise to multiple representations.

Consider the following request as an example:

(14.3) I want to eat Italian food.

While the use of the phrase *Italian food* may provide enough information for a restaurant advisor to provide reasonable recommendations, it is nevertheless quite *vague* as to what the user really wants to eat. Therefore, a vague representation of the meaning of this phrase may be appropriate for some purposes, while a more specific representation may be needed for other purposes. It will, therefore, be advantageous for a meaning representation language to support representations that maintain a certain level of vagueness. Note that it is not always easy to distinguish ambiguity from vagueness. Zwicky and Sadock (1975) provide a useful set of tests that can be used as diagnostics.

Canonical Form

The notion that single sentences can be assigned multiple meanings leads to the related phenomenon of distinct inputs that should be assigned the same meaning representation. Consider the following alternative ways of expressing example (14.1):

(14.4) Does Maharani have vegetarian dishes?

(14.5) Do they have vegetarian food at Maharani?

(14.6) Are vegetarian dishes served at Maharani?

(14.7) Does Maharani serve vegetarian fare?

Given that these alternatives use different words and have widely varying syntactic analyses, it would not be unreasonable to expect them to have

[2] This does not foreclose the use of intermediate semantic representations that maintain some level of ambiguity on the way to a single unambiguous form. Examples of such representations will be discussed in Chapter 15.

bstantially different meaning representations. Such a situation would, wever, have undesirable consequences for our matching approach to determining the truth of our representations. If the system's knowledge base ntains only a single representation of the fact in question, then the representations underlying all but one of our alternatives will fail to produce a atch. We could, of course, store all possible alternative representations of same fact in the knowledge base, but this would lead to an enormous mber of problems related to keeping such a knowledge base consistent.

The way out of this dilemma is motivated by the fact that since the answers given for each of these alternatives should be the same in all situations, we might say that they all mean the same thing, at least for the purposes of giving restaurant recommendations. In other words, at least in this domain, we can legitimately consider assigning the same meaning representation to the propositions underlying each of these requests. Taking such an approach would guarantee that our matching scheme for answering Yes-No questions will still work.

The notion that inputs that mean the same thing should have the same meaning representation is known as the doctrine of **canonical form**. This approach greatly simplifies various reasoning tasks since systems need only deal with a single meaning representation for a potentially wide range of expressions.

CANONICAL FORM

Canonical form does, of course, complicate the task of semantic analysis. To see this, note that the alternatives given above use completely different words and syntax to refer to vegetarian fare and to what restaurants do with it. More specifically, to assign the same representation to all of these requests our system will have to conclude that *vegetarian fare*, *vegetarian dishes* and *vegetarian food* refer to the same thing in this context, that the use here of *having* and *serving* are similarly equivalent, and that the different syntactic parses underlying these requests are all compatible with the same meaning representation.

Being able to assign the same representation to such diverse inputs is a tall order. Fortunately there are some systematic meaning relationships among word senses and among grammatical constructions that can be exploited to make this task tractable. Consider the issue of the meanings of the words *food*, *dish* and *fare* in these examples. A little introspection, or a glance at a dictionary, reveals that these words have a fair number of distinct uses. Fortunately, it also reveals that there is at least one sense that is shared among them all. If a system has the ability to choose that shared sense, then an identical meaning representation can be assigned to the phrases contain-

ing these words.

WORD SENSES

In general, we say that these words all have various **word senses** and that some of the senses are synonymous with one another. The process of

WORD SENSE DISAMBIGUATION

choosing the right sense in context is called **word sense disambiguation** or word sense tagging by analogy to part-of-speech tagging. The topics of synonymy, sense tagging, and a host of other topics related to word meanings will be covered in Chapters 16 and 17. Suffice it to say here that the fact that inputs may use different words does not preclude the assignment of identical meanings to them.

Just as there are systematic relationships among the meanings of different words, there are similar relationships related to the role that syntactic analyses play in assigning meanings to sentences. Specifically, alternative syntactic analyses often have meanings that are, if not identical, at least systematically related to one another. Consider the following pair of examples:

(14.8) Maharani serves vegetarian dishes.

(14.9) Vegetarian dishes are served by Maharani.

Despite the different placement of the arguments to *serve* in these examples, we can still assign *Maharani* and *vegetarian dishes* to the same roles in both of these examples because of our knowledge of the relationship between active and passive sentence constructions. In particular, we can use knowledge of where grammatical subjects and direct objects appear in these constructions to assign *Maharani*, to the role of the server, and *vegetarian dishes* to the role of thing being served in both of these examples, despite the fact that they appear in different surface locations. The precise role of the grammar in the construction of meaning representations will be covered in Chapter 15.

Inference and Variables

Continuing with the topic of the computational purposes that meaning representations should serve, we should consider more complex requests such as the following:

(14.10) Can vegetarians eat at Maharani?

Here, it would be a mistake to invoke canonical form to force our system to assign the same representation to this request as for the previous examples. The fact that this request results in the same answer as the others arises not because they mean the same thing, but because there is a commonsense connection between what vegetarians eat and what vegetarian restaurants serve. This is a fact about the world and not a fact about any particular kind of

linguistic regularity. This implies that no approach based on canonical form and simple matching will give us an appropriate answer to this request. What is needed is a systematic way to connect the meaning representation of this request with the facts about the world as they are represented in a knowledge base.

INFERENCE

We will use the term **inference** to refer generically to a system's ability to draw valid conclusions based on the meaning representation of inputs and its store of background knowledge. It must be possible for the system to draw conclusions about the truth of propositions that are not explicitly represented in the knowledge base, but are nevertheless logically derivable from the propositions that are present.

Now consider the following somewhat more complex request:

(14.11) I'd like to find a restaurant where I can get vegetarian food.

Unlike our previous examples, this request does not make reference to any particular restaurant. The user is stating that they would like information about an unknown and unnamed entity that is a restaurant that serves vegetarian food. Since this request does not mention any particular restaurant, the kind of simple matching-based approach we have been advocating is not going to work. Rather, answering this request requires a more complex kind of matching that involves the use of variables. We can gloss a representation containing such variables as follows:

$$Serves(x, VegetarianFood)$$

Matching such a proposition succeeds only if the variable x can be replaced by some known object in the knowledge base in such a way that the entire proposition will then match. The concept that is substituted for the variable can then be used to fulfill the user's request. Of course, this simple example only hints at the issues involved in the use of such variables. Suffice it to say that linguistic inputs contain many instances of all kinds of indefinite references and it is therefore critical for any meaning representation language to be able to handle this kind of expression.

Expressiveness

Finally, to be useful a meaning representation scheme must be expressive enough to handle an extremely wide range of subject matter. The ideal situation, of course, would be to have a single meaning representation language that could adequately represent the meaning of any sensible natural language utterance. Although this is probably too much to expect from any

single representational system, Section 14.3 will show that First Order Predicate Calculus is expressive enough to handle quite a lot of what needs to be represented.

14.2 Meaning Structure of Language

MEANING STRUCTURE OF LANGUAGE

The previous section focused on some of the purposes that meaning representations must serve, without saying much about what we will call the **meaning structure of language**. By this, we have in mind the various methods by which human languages convey meaning. These include a variety of conventional form-meaning associations, word-order regularities, tense systems, conjunctions and quantifiers, and a fundamental predicate-argument structure. The remainder of this section focuses exclusively on this last notion of a predicate-argument structure, which is the mechanism that has had the greatest practical influence on the nature of meaning representation languages. The remaining topics will be addressed in Chapter 15 where the primary focus will be on how they contribute to how meaning representations are assembled, rather than on the nature of the representations.

Predicate-Argument Structure

It appears to be the case that all human languages have a form of predicate-argument arrangement at the core of their semantic structure. To a first approximation, this predicate-argument structure asserts that specific relationships hold among the various concepts underlying the constituent words and phrases that make up sentences. It is largely this underlying structure that permits the creation of a single composite meaning representation from the meanings of the various parts of an input. One of the most important jobs of a grammar is to help organize this predicate-argument structure. Correspondingly, it is critical that our meaning representation languages support the predicate-argument structures presented to us by language.

We have already seen the beginnings of this concept in our discussion of verb complements in Chapters 9 and 11. There we saw that verbs dictate specific constraints on the number, grammatical category, and location of the phrases that are expected to accompany them in syntactic structures. To briefly review this idea, consider the following examples:

(14.12) I want Italian food.

(14.13) I want to spend less than five dollars.

(14.14) I want it to be close by here.

These examples can be classified as having one of the following three syntactic argument frames:

NP want NP

NP want Inf-VP

NP want NP Inf-VP

These syntactic frames specify the number, position and syntactic category of the arguments that are expected to accompany a verb. For example, the frame for the variety of *want* that appears in example (14.12) specifies the following facts:

- There are two arguments to this predicate.
- Both arguments must be *NP*s.
- The first argument is pre-verbal and plays the role of the subject.
- The second argument is post-verbal and plays the role of the direct object.

As we have shown in previous chapters, this kind of information is quite valuable in capturing a variety of important facts about syntax. By analyzing easily observable semantic information associated with these frames, we can also gain considerable insight into our meaning representations. We will begin by considering two extensions of these frames into the semantic realm: semantic roles and semantic restrictions on these roles.

The notion of a semantic role can be understood by looking at the similarities among the arguments in examples (14.12) through (14.14). In each of these cases, the pre-verbal argument always plays the role of the entity doing the wanting, while the post-verbal argument plays the role of the concept that is *wanted*. By noticing these regularities and labeling them accordingly, we can associate the surface arguments of a verb with a set of discrete roles in its underlying semantics. More generally, we can say that verb subcategorization frames allow the **linking** of arguments in the surface structure with the semantic roles these arguments play in the underlying semantic representation of an input. The study of roles associated with specific verbs and across classes of verbs is usually referred to as **thematic role** or **case role** analysis and will be studied in more detail in Section 14.4 and Chapter 16.

LINKING

THEMATIC ROLE

CASE ROLE

The notion of semantic restrictions arises directly from these semantic roles. Returning to examples (14.12) through (14.14), we can see that it is not merely the case that each initial noun phrase argument will be the *wanter* but that only certain kinds, or *categories*, of concepts can play the role of

wanter in any straightforward manner. Specifically, *want* restricts the constituents appearing as the first argument to those whose underlying concepts can actually partake in a wanting. Traditionally, this notion is referred to as a **selectional restriction**. Through the use of these selectional restrictions, verbs can specify semantic restrictions on their arguments.

SELECTIONAL RESTRICTION

Before leaving this topic, we should note that verbs are by no means the only objects in a grammar that can carry a predicate-argument structure. Consider the following phrases from the BERP corpus:

(14.15) an Italian restaurant under fifteen dollars

In this example, the meaning representation associated with the preposition *under* can be seen as having something like the following structure:

$$Under(ItalianRestaurant, \$15)$$

In other words, prepositions can be characterized as two-argument predicates where the first argument is an object that is being placed in some relation to the second argument.

Another non-verb based predicate-argument structure is illustrated in the following example:

(14.16) Make a reservation for this evening for a table for two persons at 8.

Here, the predicate-argument structure is based on the concept underlying the noun *reservation*, rather than *make*, the main verb in the phrase. This example gives rise to a four argument predicate structure like the following:

$$Reservation(Hearer, Today, 8PM, 2)$$

This discussion makes it clear that any useful meaning representation language must be organized in a way that supports the specification of semantic predicate-argument structures. Specifically, it must include support for the kind of semantic information that languages present:

- variable arity predicate-argument structures
- the semantic labeling of arguments to predicates
- the statement of semantic constraints on the fillers of argument roles

14.3 First Order Predicate Calculus

First Order Predicate Calculus (FOPC) is a flexible, well-understood, and computationally tractable approach to the representation of knowledge that satisfies many of the requirements raised in Sections 14.1 and 14.2 for a meaning representation language. Specifically, it provides a sound computational basis for the verifiability, inference, and expressiveness requirements. However, the most attractive feature of FOPC is the fact that it makes very few specific commitments as to how things ought to be represented. As we will see, the specific commitments it does make are ones that are fairly easy to live with; the represented world consists of objects, properties of objects, and relations among objects.

The remainder of this section first provides an introduction to the basic syntax and semantics of FOPC and then describes the application of FOPC to a number of linguistically relevant topics. Section 14.6 then discusses the connections between FOPC and some of the other representations shown earlier in Figure 14.1.

Elements of FOPC

We will explore FOPC in a bottom-up fashion by first examining its various atomic elements and then showing how they can be composed to create larger meaning representations. Figure 14.2, which provides a complete context-free grammar for the particular syntax of FOPC that we will be using, will be our roadmap for this section.

Let's begin by examining the notion of a **Term**, the FOPC device for representing objects. As can be seen from Figure 14.2, FOPC provides three ways to represent these basic building blocks: constants, functions, and variables. Each of these devices can be thought of as a way of naming, or pointing to, an object in the world under consideration.

TERM

Constants in FOPC refer to specific objects in the world being described. Such constants are conventionally depicted as either single capitalized letters such as *A* and *B* or single capitalized words that are often reminiscent of proper nouns such as *Maharani* and *Harry*. Like programming language constants, FOPC constants refer to exactly one object. Objects can, however, have multiple constants that refer to them.

CONSTANTS

Functions in FOPC correspond to concepts that are often expressed in English as genitives such as *the location of Maharani* or *Maharani's loca-*

FUNCTIONS

$$
\begin{aligned}
\textit{Formula} &\rightarrow \textit{AtomicFormula} \\
&\mid \textit{Formula Connective Formula} \\
&\mid \textit{Quantifier Variable}, \ldots \textit{Formula} \\
&\mid \neg \textit{Formula} \\
&\mid (\textit{Formula}) \\
\\
\textit{AtomicFormula} &\rightarrow \textit{Predicate}(\textit{Term}, \ldots) \\
\\
\textit{Term} &\rightarrow \textit{Function}(\textit{Term}, \ldots) \\
&\mid \textit{Constant} \\
&\mid \textit{Variable} \\
\\
\textit{Connective} &\rightarrow \wedge \mid \vee \mid \Rightarrow \\
\textit{Quantifier} &\rightarrow \forall \mid \exists \\
\textit{Constant} &\rightarrow A \mid \textit{VegetarianFood} \mid \textit{Maharani} \cdots \\
\textit{Variable} &\rightarrow x \mid y \mid \cdots \\
\textit{Predicate} &\rightarrow \textit{Serves} \mid \textit{Near} \mid \cdots \\
\textit{Function} &\rightarrow \textit{LocationOf} \mid \textit{CuisineOf} \mid \cdots
\end{aligned}
$$

Figure 14.2 A context-free grammar specification of the syntax of First Order Predicate Calculus representations. Adapted from Russell and Norvig (1995).

tion. A FOPC translation of such an expression might look like the following.

$$\textit{LocationOf}(\textit{Maharani})$$

FOPC functions are syntactically the same as single argument predicates. It is important to remember, however, that while they have the appearance of predicates they are in fact *Terms* in that they refer to unique objects. Functions provide a convenient way to refer to specific objects without having to associate a named constant with them. This is particularly convenient in cases where many named objects, like restaurants, will have a unique concept such as a location associated with them.

VARIABLE

The notion of a **variable** is our final FOPC mechanism for referring to

objects. Variables, which are normally depicted as single lower-case letters, give us the ability to make assertions and draw inferences about objects without having to make reference to any particular named object. This ability to make statements about anonymous objects comes in two flavors: making statements about a particular unknown object and making statements about all the objects in some arbitrary world of objects. We will return to the topic of variables after we have presented quantifiers, the elements of FOPC that will make them useful.

Now that we have the means to refer to objects, we can move on to the FOPC mechanisms that are used to state relations that hold among objects. As one might guess from its name, FOPC is organized around the notion of the predicate. Predicates are symbols that refer to, or name, the relations that hold among some fixed number of objects in a given domain. Returning to the example introduced informally in Section 14.1, a reasonable FOPC representation for *Maharani serves vegetarian food* might look like the following formula:

$$\textit{Serves}(\textit{Maharani}, \textit{VegetarianFood})$$

This FOPC sentence asserts that *Serves*, a two-place predicate, holds between the objects denoted by the constants *Maharani* and *VegetarianFood*.

A somewhat different use of predicates is illustrated by the following typical representation for a sentence like *Maharani is a restaurant*:

$$\textit{Restaurant}(\textit{Maharani})$$

This is an example of a one-place predicate that is used, not to relate multiple objects, but rather to assert a property of a single object. In this case, it encodes the category membership of *Maharani*. We should note that while this is a commonplace way to deal with categories it is probably not the most useful. Section 14.4 will return to the topic of the representation of categories.

With the ability to refer to objects, to assert facts about objects, and to relate objects to one another, we have the ability to create rudimentary composite representations. These representations correspond to the atomic formula level in Figure 14.2. Recall that this ability to create composite meaning representations was one of the core components of the meaning structure of language described in Section 14.2.

This ability to compose complex representations is not limited to the use of single predicates. Larger composite representations can also be put together through the use of **logical connectives**. As can be seen from Figure 14.2, logical connectives give us the ability to create larger representations

LOGICAL CONNECTIVES

by conjoining logical formulas using one of three operators. Consider for example, the following BERP sentence and one possible representation for it:

(14.17) I only have five dollars and I don't have a lot of time.

$$Have(Speaker, FiveDollars) \wedge \neg Have(Speaker, LotOfTime)$$

The semantic representation for this example is built up in a straightforward way from semantics of the individual clauses through the use of the $\wedge$ and $\neg$ operators. Note that the recursive nature of the grammar in Figure 14.2 allows an infinite number of logical formulas to be created through the use of these connectives. Thus as with syntax, we have the ability to create an infinite number of representations using a finite device.

The Semantics of FOPC

The various objects, properties, and relations represented in a FOPC knowledge base acquire their meanings by virtue of their correspondence to objects, properties, and relations out in the external world being modeled by the knowledge base. FOPC sentences can, therefore, be assigned a value of *True* or *False* based on whether the propositions they encode are in accord with the world or not.

Consider the following example:

(14.18) Ay Caramba is near ICSI.

Capturing the meaning of this example in FOPC involves identifying the *Terms* and *Predicates* that correspond to the various grammatical elements in the sentence, and creating logical formulas that capture the relations implied by the words and syntax of the sentence. For this example, such an effort might yield something like the following:

$$Near(LocationOf(AyCaramba), LocationOf(ICSI))$$

The meaning of this logical formula then arises from the relationship between the terms $LocationOf(AyCaramba)$, $LocationOf(ICSI)$, the predicate *Near*, and the objects and relation they correspond to in the world being modeled. Specifically, this sentence can be assigned a value of *True* or *False* based on whether or not the real Ay Caramba is actually close to ICSI or not. Of course, since our computers rarely have direct access to the outside world we have to rely on some other means to determine the truth of formulas like this one.

For our current purposes, we will adopt what is known as a database semantics for determining the truth of our logical formulas. Operationally,

atomic formulas are taken to be true if they are literally present in the knowledge base or if they can be inferred from other formula that are in the knowledge base. The interpretations of formulas involving logical connectives is based on the meaning of the components in the formulas combined with the meanings of the connectives they contain. Figure 14.3 gives interpretations for each of the logical operators shown in Figure 14.2.

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
False	*False*	*True*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*	*True*
True	*False*	*False*	*False*	*True*	*False*
True	*True*	*False*	*True*	*True*	*True*

Figure 14.3 Truth table giving the semantics of the various logical connectives.

The semantics of the $\wedge$ (and), and $\neg$ (not) operators are fairly straightforward, and are correlated with at least some of the senses of their corresponding English terms. However, it is worth pointing out that the $\vee$ (or) operator is not disjunctive in the same way that the corresponding English word is, and that the $\Rightarrow$ (implies) operator is only loosely based on any commonsense notions of implication or causation. As we will see in more detail in Section 14.4, in most cases it is safest to rely directly on the entries in the truth table, rather than on intuitions arising from the names of the operators.

Variables and Quantifiers

We now have all the machinery necessary to return to our earlier discussion of variables. As noted above, variables are used in two ways in FOPC: to refer to particular anonymous objects and to refer generically to all objects in a collection. These two uses are made possible through the use of operators known as **quantifiers**. The two operators that are basic to FOPC are the existential quantifier, which is denoted $\exists$, and is pronounced as "there exists", and the universal quantifier, which is denoted $\forall$, and is pronounced as "for all". QUANTIFIERS

The need for an existentially quantified variable is often signaled by the presence of an indefinite noun phrase in English. Consider the following example:

(14.19) a restaurant that serves Mexican food near ICSI.

Here reference is being made to an anonymous object of a specified category with particular properties. The following would be a reasonable representation of the meaning of such a phrase:

$$\exists x Restaurant(x) \\ \wedge Serves(x, MexicanFood) \\ \wedge Near((LocationOf(x), LocationOf(ICSI))$$

The existential quantifier at the head of this sentence instructs us on how to interpret the variable x in the context of this sentence. Informally, it says that for this sentence to be true there must be at least one object such that if we were to substitute it for the variable x, the resulting sentence would be true. For example, if *AyCaramba* is a Mexican restaurant near ICSI, then substituting *AyCaramba* for x results in the following logical formula:

$$Restaurant(AyCaramba) \\ \wedge Serves(AyCaramba, MexicanFood) \\ \wedge Near((LocationOf(AyCaramba), LocationOf(ICSI))$$

Based on the semantics of the $\wedge$ operator, this sentence will be true if all of its three component atomic formulas are true. These in turn will be true if they are either present in the system's knowledge base or can be inferred from other facts in the knowledge base.

The use of the universal quantifier also has an interpretation based on substitution of known objects for variables. The substitution semantics for the universal quantifier takes the expression *for all* quite literally; the $\forall$ operator states that for the logical formula in question to be true the substitution of *any* object in the knowledge base for the universally quantified variable should result in a true formula. This is in marked contrast to the $\exists$ operator which only insists on a single valid substitution for the sentence to be true.

Consider the following example:

(14.20) All vegetarian restaurants serve vegetarian food.

A reasonable representation for this sentence would be something like the following:

$$\forall x VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$$

For this sentence to be true, it must be the case that every substitution of a known object for x must result in a sentence that is true. We can divide up the set of all possible substitutions into the set of objects consisting of vegetarian restaurants and the set consisting of everything else. Let us first consider the

case where the substituted object actually is a vegetarian restaurant; one such substitution would result in the following sentence:

$$VegetarianRestaurant(Maharani) \\ \Rightarrow Serves(Maharani, VegetarianFood)$$

If we assume that we know that the consequent clause,

$$Serves(Maharani, VegetarianFood)$$

is true then this sentence as a whole must be true. Both the antecedent and the consequent have the value *True* and, therefore, according to the first two rows of Figure 14.3 the sentence itself can have the value *True*. This result will, of course, be the same for all possible substitutions of *Terms* representing vegetarian restaurants for x.

Remember, however, that for this sentence to be true it must be true for all possible substitutions. What happens when we consider a substitution from the set of a objects that are not vegetarian restaurants? Consider the substitution of a non-vegetarian restaurant such as *Ay Caramba's* for the variable x:

$$VegetarianRestaurant(AyCaramba) \\ \Rightarrow Serves(AyCaramba, VegetarianFood)$$

Since the antecedent of the implication is *False*, we can determine from Figure 14.3 that the sentence is always *True*, again satisfying the $\forall$ constraint.

Note, that it may still be the case that *Ay Caramba* serves vegetarian food without actually being a vegetarian restaurant. Note also, that despite our choice of examples, there are no implied categorical restrictions on the objects that can be substituted for x by this kind of reasoning. In other words, there is no restriction of x to restaurants or concepts related to them. Consider the following substitution:

$$VegetarianRestaurant(Carburetor) \\ \Rightarrow Serves(Carburetor, VegetarianFood)$$

Here the antecedent is still false and hence the rule remains true under this kind of irrelevant substitution.

To review, variables in logical formulas must be either existentially ($\exists$) or universally ($\forall$) quantified. To satisfy an existentially quantified variable, there must be at least one substitution that results in a true sentence. Sentences with universally quantified variables must be true under all possible substitutions.

Inference

One of the most important desiderata given in Section 14.1 for a meaning representation language is that it should support inference—the ability to add valid new propositions to a knowledge base, or to determine the truth of propositions not explicitly contained within a knowledge base. This section briefly discusses **modus ponens**, the most important inference method provided by FOPC. Applications of modus ponens will be discussed in Chapter 18.

MODUS PONENS

Modus ponens is a familiar form of inference that corresponds to what is informally known as *if-then* reasoning. We can abstractly define modus ponens as follows, where α and β should be taken as FOPC formulas:

$$\frac{\alpha \quad \alpha \Rightarrow \beta}{\beta}$$

In general, schemas like this indicate that the formula below the line can be inferred from the formulas above the line by some form of inference. Modus ponens simply states that if the left-hand side of an implication rule is present in the knowledge base, then the right-hand side of the rule can be inferred. In the following discussions, we will refer to the left-hand side of an implication as the antecedent, and the right-hand side as the consequent.

As an example of a typical use of modus ponens, consider the following example, which uses a rule from the last section:

$$\frac{VegetarianRestaurant(Rudys) \quad \forall x VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)}{Serves(Rudys, VegetarianFood)} \tag{14.21}$$

Here, the formula $VegetarianRestaurant(Rudys)$ matches the antecedent of the rule, thus allowing us to use modus ponens to conclude $Serves(Rudys, VegetarianFood)$.

FORWARD CHAINING

Modus ponens is typically put to practical use in one of two ways: forward chaining and backward chaining. In **forward chaining** systems, modus ponens is used in precisely the manner just described. As individual facts are added to the knowledge base, modus ponens is used to fire all applicable implication rules. In this kind of arrangement, as soon as a new fact is added to the knowledge base, all applicable implication rules are found and applied, each resulting in the addition new facts to the knowledge base. These new

positions in turn can be used to fire implication rules applicable to them. process continues until no further facts can be deduced.

The forward chaining approach has the advantage that facts will be nt in the knowledge base when needed, since in a sense all inference formed in advance. This can substantially reduce the time needed to wer subsequent queries since they should all amount to simple lookups. disadvantage of this approach is that facts may be inferred and stored will never be needed. **Production systems**, which are heavily used PRODUCTION SYSTEMS cognitive modeling work, are forward chaining inference systems augted with additional control knowledge that governs which rules are to be ed.

In **backward chaining**, modus ponens is run in reverse to prove spe- BACKWARD CHAINING cific propositions, called queries. The first step is to see if the query formula is true by determining if it is present in the knowledge base. If it is not, then the next step is to search for applicable implication rules present in the knowledge base. An applicable rule is one where the consequent of the rule matches the query formula. If there are such any such rules, then the query can be proved if the antecedent of any one them can be shown to be true. Not surprisingly, this can be performed recursively by backward chaining on the antecedent as a new query. The **Prolog** programming language is a backward chaining system that implements this strategy.

To see how this works, let's assume that we have been asked to verify the truth of the proposition $Serves(Rudys, VegetarianFood)$, assuming the facts given above the line in (14.21). Since it is not present in the knowledge base, a search for an applicable rule is initiated that results in the rule given above. After substituting, the constant *Rudys* for the variable x, our next task is to prove the antecedent of the rule, $VegetarianRestaurant(Rudys)$, which of course is one of the facts we are given.

Note that it is critical to distinguish between reasoning via backward chaining from queries to known facts, and reasoning backwards from known consequents to unknown antecedents. To be specific, by reasoning backwards we mean that if the consequent of a rule is known to be true, we assume that the antecedent will be as well. For example, let's assume that we know that $Serves(Rudys, VegetarianFood)$ is true. Since this fact matches the consequent of our rule, we might reason backwards to the conclusion that $VegetarianRestaurant(Rudys)$.

While backward chaining is a sound method of reasoning, reasoning backwards is an invalid, though frequently useful, form of *plausible reasoning*. Plausible reasoning from consequents to antecedents is known as

ABDUCTION **abduction**, and as we will see in Chapter 18 is often useful in accounting fo many of the inferences people make while analyzing extended discourses.

COMPLETE While forward and backward reasoning are sound, neither is **complete**. This means that there are valid inferences that can not be found by systems using these methods alone. Fortunately, there is an alternative inference technique called **resolution** that is sound and complete. Unfortunately, inference systems based on resolution are far more computationally expensive than forward or backward chaining systems. In practice, therefore, most systems use some form of chaining, and place a burden on knowledge base developers to encode the knowledge in a fashion that permits the necessary inferences to be drawn.

RESOLUTION

14.4 Some Linguistically Relevant Concepts

Entire lives have been spent studying the representation of various aspects of human knowledge. These efforts have ranged from tightly focused efforts to represent individual domains such as time, to monumental efforts to encode all of our commonsense knowledge of the world (Lenat and Guha, 1991). Our focus here is considerably more modest. This section provides a brief overview of the representation of a few important topics that have clear implications for language processing. Specifically, the following sections provide introductions to the meaning representations of categories, events, time, and beliefs.

Categories

As we noted in Section 14.2, words with predicate-like semantics often express preferences for the semantics of their arguments in the form of selectional restrictions. These restrictions are typically expressed in the form of semantically-based categories where all the members of a category share a set of relevant features.

The most common way to represent categories is to create a unary predicate for each category of interest. Such predicates can then be asserted for each member of that category. For example, in our restaurant discussions we have been using the unary predicate *VegetarianRestaurant* as in:

$$VegetarianRestaurant(Maharani)$$

Similar logical formulas would be included in our knowledge base for each known vegetarian restaurant.

Unfortunately, in this method categories are relations, rather than full-ged objects. It is, therefore, difficult to make assertions about categories iselves, rather than about their individual members. For example, we ght want to designate the most popular member of a given category as in following expression:

$$MostPopular(Maharani, VegetarianRestaurant)$$

nfortunately, this is not a legal FOPC formula since the arguments to pred-tes in FOPC must be *Terms*, not other predicates.

One way to solve this problem is to represent all the concepts that want to make statements about as full-fledged objects via a tech-ique called **reification**. In this case, we can represent the category of *getarianRestaurant* as an object just as *Maharani* is. The notion of mem-rship in such a category is then denoted via a membership relation as in e following:

REIFICATION

$$ISA(Maharani, VegetarianRestaurant)$$

The relation denoted by *ISA* (is a) holds between objects and the cate-ories in which they are members. This technique can be extended to create ierarchies of categories through the use of other similar relations, as in the ollowing:

$$AKO(VegetarianRestaurant, Restaurant)$$

Here, the relation *AKO* (a kind of) holds between categories and denotes category inclusion relationship. Of course, to truly give these predicates eaning they would have to be situated in a larger set of facts defining cate-gories as sets.

Chapter 16 discusses the practical use of such relations in databases of lexical relations, in the representation of selectional restrictions, and in word sense disambiguation.

Events

The representations for events that we have used until now have consisted of single predicates with as many arguments as are needed to incorporate all the roles associated with a given example. For example, the representation for *making a reservation* discussed in Section 14.2 consisted of a single predicate with arguments for the person making the reservation, the restaurant, the day, the time, and the number of people in the party, as in the following:

$$Reservation(Hearer, Maharani, Today, 8PM, 2)$$

In the case of verbs, this approach simply assumes that the predicate representing the meaning of a verb has the same number of arguments as are present in the verb's syntactic subcategorization frame.

Unfortunately, there are three problems with this approach that make it awkward to apply in practice:

- determining the correct number of roles for any given event
- representing facts about the roles associated with an event
- ensuring that all the correct inferences can be derived directly from the representation of an event
- ensuring that no incorrect inferences can be derived from the representation of an event

We will explore these, and other related issues, by considering a series of representations for events. This discussion will focus on the following examples of the verb *eat*:

(14.22) I ate.

(14.23) I ate a turkey sandwich.

(14.24) I ate a turkey sandwich at my desk.

(14.25) I ate at my desk.

(14.26) I ate lunch.

(14.27) I ate a turkey sandwich for lunch.

(14.28) I ate a turkey sandwich for lunch at my desk.

Clearly, the variable number of arguments for a predicate-bearing verb like *eat* poses a tricky problem. While we would like to think that all of these examples denote the same kind of event, predicates in FOPC have fixed **arity**—they take a fixed number of arguments. ARITY

One possible solution is suggested by the way that examples like these are handled syntactically. The solution given in Chapter 11 was to create one subcategorization frame for each of the configurations of arguments that a verb allows. The semantic analog to this approach is to create as many different *eating* predicates as are needed to handle all of the ways that *eat* behaves. Such an approach would yield the following kinds of representations for examples (14.22) through (14.22).

$$Eating_1(Speaker)$$
$$Eating_2(Speaker, TurkeySandwich)$$
$$Eating_3(Speaker, TurkeySandwich, Desk)$$
$$Eating_4(Speaker, Desk)$$
$$Eating_5(Speaker, Lunch)$$
$$Eating_6(Speaker, TurkeySandwich, Lunch)$$
$$Eating_7(Speaker, TurkeySandwich, Lunch, Desk)$$

This approach simply sidesteps the issue of how many arguments the *Eating* predicate should have by creating distinct predicates for each of the subcategorization frames. Unfortunately, this approach comes at a rather high cost. Other than the suggestive names of the predicates, there is nothing to tie these events to one another even though there are obvious logical relations among them. Specifically, if example (14.28) is true then all of the other examples are true as well. Similarly, if example (14.27) is true then examples (14.22), (14.23), and (14.26) must also be true. Such logical connections can not be made on the basis of these predicates alone. Moreover, we would expect a commonsense knowledge base to contain logical connections between concepts like *Eating* and related concepts like *Hunger* and *Food*.

One method to solve these problems involves the use of what are called **meaning postulates**. Consider the following example postulate: MEANING POSTULATES

$$\forall w, x, y, z \; Eating_7(w, x, y, z) \Rightarrow Eating_6(w, x, y)$$

This postulate explicitly ties together the semantics of two of our predicates. Other postulates could be created to handle the rest of the logical relations among the various *Eatings* and the connections from them to other related concepts.

Although such an approach might be made to work in small domains, it clearly has scalability problems. A somewhat more sensible approach is to say that examples (14.22) through (14.28) all reference the same predicate with some of the arguments missing from some of the surface forms. Under this approach, as many arguments are included in the definition of the predicate as ever appear with it in an input. Adopting the structure of a predicate like $Eating_7$ as an example would give us a predicate with four arguments denoting the eater, thing eaten, meal being eaten and the location of the eating. The following formulas would then capture the semantics of

our examples:

$\exists w,x,y\ Eating(Speaker,w,x,y)$
$\exists w,x\ Eating(Speaker,TurkeySandwich,w,x)$
$\exists w\ Eating(Speaker,TurkeySandwich,w,Desk)$
$\exists w,x\ Eating(Speaker,w,x,Desk)$
$\exists w,x\ Eating(Speaker,w,Lunch,x)$
$\exists w\ Eating(Speaker,TurkeySandwich,Lunch,w)$
$Eating(Speaker,TurkeySandwich,Lunch,Desk)$

This approach directly yields the obvious logical connections among these formulas without the use of meaning postulates. Specifically, all of the sentences with ground terms as arguments logically imply the truth of the formulas with existentially bound variables as arguments.

Unfortunately, this approach still has at least two glaring deficiencies: it makes too many commitments, and it does not let us individuate events. As an example of how it makes too many commitments, consider how we accommodated the *for lunch* complement in examples (14.26) through (14.28): a third argument, the meal being eaten, was added to the *Eating* predicate. The presence of this argument implicitly makes it the case that all eating events are associated with a meal (i.e., breakfast, lunch, or dinner). More specifically, the existentially quantified variable for the meal argument in the above examples states that there is some formal meal associated with each of these eatings. This is clearly silly since one can certainly eat something independent of it being associated with a meal.

To see how this approach fails to properly individuate events, consider the following formulas.

$\exists w,x\ Eating(Speaker,w,x,Desk)$
$\exists w,x\ Eating(Speaker,w,Lunch,x)$
$\exists w,x\ Eating(Speaker,w,Lunch,Desk)$

If we knew that the first two formula were referring to the same event, they could be combined to create the third representation. Unfortunately, with the current representation we have no way of telling if this is possible. The independent facts that *I ate at my desk* and *I ate lunch* do not permit us to conclude that *I ate lunch at my desk*. Clearly what is lacking is some way of referring to the events in question.

As with categories, we can solve these problems if we employ reification to elevate events to objects that can be quantified and related to a other objects via sets of defined relations (Davidson, 1967; Parsons, 1990). Con-

sider the representation of example (14.23) under this kind of approach.

$\exists w\ ISA(w,Eating)$
$\quad \wedge Eater(w,Speaker) \wedge Eaten(w,TurkeySandwich)$

This representation states that there is an eating event where the *Speaker* is doing the eating and a *TurkeySandwich* is being eaten. The meaning representations for examples (14.22) and (14.27) can be constructed similarly.

$\exists w\ ISA(w,Eating) \wedge Eater(w,Speaker)$
$\exists w\ ISA(w,Eating)$
$\quad \wedge Eater(w,Speaker) \wedge Eaten(w,TurkeySandwich)$
$\quad \wedge MealEaten(w,Lunch)$

Under this reified-event approach:

- There is no need to specify a fixed number of arguments for a given surface predicate, rather as many roles and fillers can be glued on as appear in the input.
- No more roles are postulated than are mentioned in the input.
- The logical connections among closely related examples is satisfied without the need for meaning postulates.

Representing Time

In the preceding discussion of events, we did not address the issue of representing the time when the represented events are supposed to have occurred. The representation of such information in a useful form is the domain of **temporal logic**. This discussion will serve to introduce the most basic concerns of temporal logic along with a brief discussion of the means by which human languages convey temporal information, which among other things includes **tense logic**, the ways that verb tenses convey temporal information.

TEMPORAL LOGIC

TENSE LOGIC

The most straightforward theory of time hold that it flows inexorably forward, and that events are associated with either points or intervals in time, as on a timeline. Given these notions, an ordering can be imposed on distinct events by situating them on the timeline. More specifically, we can say that one event *precedes* another, if the flow of time leads from the first event to the second. Accompanying these notions in most theories is the idea of the current moment in time. Combining this notion with the idea of a temporal ordering relationship yields the familiar notions of past, present and future.

Not surprisingly, there are a large number of schemes for representing this kind of temporal information. The one presented here is a fairly simple

one that stays within the FOPC framework of reified events that we have b pursuing. Consider the following examples:

(14.29) I arrived in New York.

(14.30) I am arriving in New York.

(14.31) I will arrive in New York.

These sentences all refer to the same kind of event and differ solely in the tense of the verb. In our current scheme for representing events, all three would share the following kind of representation, which lacks any temporal information:

$$\exists w\, ISA(w, Arriving) \\ \wedge Arriver(w, Speaker) \wedge Destination(w, NewYork)$$

The temporal information provided by the tense of the verbs can be exploited by predicating additional information about the event variable w. Specifically, we can add temporal variables representing the interval corresponding to the event, the end point of the event, and temporal predicates relating this end point to the current time as indicated by the tense of the verb. Such an approach yields the following representations for our *arriving* examples:

$$\exists i, e, w, t\, ISA(w, Arriving) \\ \wedge Arriver(w, Speaker) \wedge Destination(w, NewYork) \\ IntervalOf(w, i) \wedge EndPoint(i, e) \wedge Precedes(e, Now)$$

$$\exists i, e, w, t\, ISA(w, Arriving) \\ \wedge Arriver(w, Speaker) \wedge Destination(w, NewYork) \\ IntervalOf(w, i) \wedge MemberOf(i, Now)$$

$$\exists i, e, w, t\, ISA(w, Arriving) \\ \wedge Arriver(w, Speaker) \wedge Destination(w, NewYork) \\ IntervalOf(w, i) \wedge EndPoint(i, e) \wedge Precedes(Now, e)$$

This representation introduces a variable to stand for the interval of time associated with the event, and a variable that stands for the end of that interval. The two-place predicate *Precedes* represents the notion that the first time point argument precedes the second in time; the constant *Now* refers to the current time. For past events, the end point of the interval must precede the current time. Similarly, for future events the current time must precede the end of the event. For events happening in the present, the current time is contained within the event interval.

Unfortunately, the relation between simple verb tenses and points in time is by no means straightforward. Consider the following examples:

(14.32) Ok, we fly from San Francisco to Boston at 10.

(14.33) Flight 1390 will be at the gate an hour now.

In the first example, the present tense of the verb *fly* is used to refer to a future event, while in the second the future tense is used to refer to a past event.

More complications occur when we consider some of the other verb tenses. Consider the following examples:

(14.34) Flight 1902 arrived late.

(14.35) Flight 1902 had arrived late.

Although both refer to events in the past, representing them in the same way seems wrong. The second example seems to have another unnamed event lurking in the background (e.g., Flight 1902 had already arrived late *when* something else happened). To account for this phenomena, Reichenbach (1947) introduced the notion of a **reference point**. In our simple temporal scheme, the current moment in time is equated with the time of the utterance, and is used as a reference point for when the event occurred (before, at, or after). In Reichenbach's approach, the notion of the reference point is separated out from the utterance time and the event time. The following examples illustrate the basics of this approach:

REFERENCE POINT

(14.36) When Mary's flight departed, I ate lunch.

(14.37) When Mary's flight departed, I had eaten lunch.

In both of these examples, the eating event has happened in the past, i.e. prior to the utterance. However, the verb tense in the first example indicates that the eating event began when the flight departed, while the second example indicates that the eating was accomplished prior to the flight's departure. Therefore, in Reichenbach's terms the *departure* event specifies the reference point. These facts can be accommodated by asserting additional constraints relating the *eating* and *departure* events. In the first example, the reference point precedes the *eating* event, and in the second example, the eating precedes the reference point. Figure 14.4 illustrates Reichenbach's approach with the primary English tenses. Exercise 14.9 asks you to represent these examples in FOPC.

This discussion has focused narrowly on the broad notions of past, present, and future and how they are signaled by verb tenses. Of course,

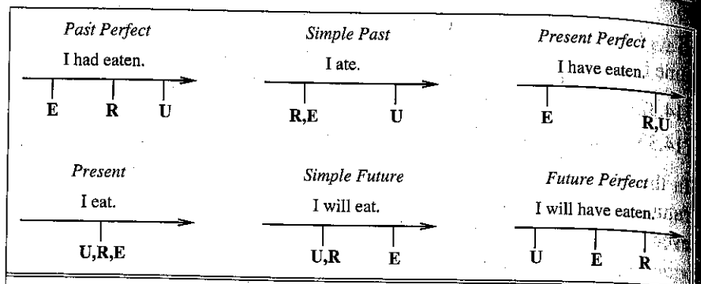


Figure 14.4 Reichenbach's approach applied to various English tenses. In these diagrams, time flows from left to right, an **E** denotes the time of the event, an **R** denotes the reference time, and an **U** denotes the time of the utterance.

languages also have many other more direct and more specific ways to convey temporal information, including the use of a wide variety of temporal expressions as in the following ATIS examples:

(14.38) I'd like to go at 6:45, in the morning.

(14.39) Somewhere around noon, please.

(14.40) Later in the afternoon, near 6PM.

As we will see in the next chapter, grammars for such temporal expressions are of considerable practical importance in information extraction and question-answering applications.

Finally, we should note that there is a systematic conceptual organization reflected in examples like these. In particular, temporal expressions in English are frequently expressed in spatial terms, as is illustrated by the various uses of *at*, *in*, *somewhere* and *near* in these examples (Lakoff and Johnson, 1980; Jackendoff, 1983) Metaphorical organizations such as these, where one domain is systematically expressed in terms of another, will be discussed in more detail in Chapter 16.

Aspect

ASPECT

In the last section, we discussed ways to represent the time of an event with respect to the time of an utterance describing it. In this section, we address the notion of **aspect**, which concerns a cluster of related topics, including whether an event has ended or is ongoing, whether it is conceptualized as happening at a point in time or over some interval, and whether or not any particular state in the world comes about because of it. Based on these and

related notions, event expressions have traditionally been divided into four general classes: **statives**, **activities**, **accomplishments**, and **achievements**. The following examples provide prototypical instances of each class.

Stative: I know my departure gate.

Activity: John is flying.

Accomplishment: Sally booked her flight.

Achievement: She found her gate.

Although the earliest versions of this classification were discussed by Aristotle, the one presented here is due to Vendler (1967). In the following discussion, we'll present a brief characterization of each of the four classes, along with some diagnostic techniques suggested in Dowty (1979) for identifying examples of each kind.

Stative expressions represent the notion of an event participant having a particular property, or being in a state, at a given point in time. As such, they can be thought of as capturing an aspect of a world at a single point in time. Consider the following ATIS examples. STATIVE EXPRESSIONS

(14.41) I like Flight 840 arriving at 10:06.

(14.42) I need the cheapest fare.

(14.43) I have a round trip ticket for $662.

(14.44) I want to go first class.

In examples like these, the event participant denoted by the subject can be seen as experiencing something at a specific point in time. Whether or not the experiencer was in the same state earlier, or will be in the future is left unspecified.

There are a number of diagnostic tests for identifying statives. As an example, stative verbs are distinctly odd when used in the progressive form.

(14.45) *I am needing the cheapest fare on this day.

(14.46) *I am wanting to go first class.

We should note that in these and subsequent examples, we are using an * to indicate a broadened notion of ill-formedness that may include both semantic and syntactic factors.

Statives are also odd when used as imperatives.

(14.47) *Need the cheapest fare!

Finally, statives are not easily modified by adverbs like *deliberately* and *carefully*.

(14.48) *I deliberately like Flight 840 arriving at 10:06.

(14.49) *I carefully like Flight 840 arriving at 10:06.

ACTIVITY EXPRESSIONS

Activity expressions describe events undertaken by a participant that have no particular end point. Unlike statives, activities are seen as occurring over some span of time, and are therefore not associated with single points in time. Consider the following examples:

(14.50) She drove a Mazda.

(14.51) I live in Brooklyn.

These examples both specify that the subject is engaged in, or has engaged in, the activity specified by the verb for some period of time.

Unlike statives, activity expressions are fine in both the progressive and imperative forms.

(14.52) She is living in Brooklyn.

(14.53) Drive a Mazda!

However, like statives, activity expressions are odd when temporally modified with temporal expressions using *in*.

(14.54) *I live in Brooklyn in a month.

(14.55) *She drove a Mazda in an hour.

They can, however, successfully be used with *for* temporal adverbials, as in the following examples:

(14.56) I live in Brooklyn for a month.

(14.57) She drove a Mazda for an hour.

ACCOMPLISHMENT EXPRESSIONS

Unlike activities, **accomplishment expressions** describe events that have a natural end point and result in a particular state. Consider the following examples:

(14.58) He booked me a reservation.

(14.59) United flew me to New York.

In these examples, there is an event that is seen as occurring over some period of time that ends when the intended state is accomplished.

A number of diagnostics can be used to distinguish accomplishment events from activities. Consider the following examples, which make use of the word *stop* as a test.

(14.60) I stopped living in Brooklyn.

(14.61) She stopped booking my flight.

first example, which is an activity, one can safely conclude that the ment *I lived in Brooklyn* even though this activity came to an end. How- from the second example, one can not conclude the statement *She ed her flight*, since the activity was stopped before the intended state was mplished. Therefore, although stopping an activity entails that the ac- took place, stopping an accomplishment event indicates that the event ot succeed.

Activities and accomplishments can also be distinguished by how they be modified by various temporal adverbials. Consider the following mples:

62) *I lived in Brooklyn in a year.

63) She booked a flight in a minute.

general, accomplishments can be modified by *in* temporal expressions, ile simple activities can not.

ACHIEVEMENT EXPRESSIONS

The final aspectual class, **achievement expressions**, are similar to ac- mplishments in that they result in a state. Consider the following exam- les:

(14.64) She found her gate.

(14.65) I reached New York.

Unlike accomplishments, achievement events are thought of as happening in an instant, and are not equated with any particular activity leading up to the state. To be more specific, the events in these examples may have been preceded by extended *searching* or *traveling* events, but the events corresponding directly to *found* and *reach* are conceived of as points not intervals.

The point-like nature of these events has implications for how they can be temporally modified. In particular, consider the following examples:

(14.66) I lived in New York for a year.

(14.67) *I reached New York for a few minutes.

Unlike activity and accomplishment expressions, achievements can not be modified by *for* adverbials.

Achievements can also be distinguished from accomplishments by employing the word *stop*, as we did earlier. Consider the following examples:

(14.68) I stopped booking my flight.

(14.69) *I stopped reaching New York.

As we saw earlier, using *stop* with an accomplishment expression results in a failure to reach the intended state. Note, however, that the resulting

expression is perfectly well-formed. On the other hand, using *stop* with ac
achievement example is unacceptable.

We should note that since both accomplishments and achievements are
events that result in a state, they are sometimes characterized as sub-types of
a single aspectual class. Members of this combined class are known as telic
TELIC EVENTUALITIES **eventualities**.

Before moving on, we should make two points about this classification
scheme. The first point is that event expressions can easily be shifted from
one class to another. Consider the following examples:

(14.70) I flew.

(14.71) I flew to New York.

The first example is a simple activity; it has no natural end point and can
not be temporally modified by *in* temporal expressions. On the other hand,
the second example is clearly an accomplishment event since it has an end
point, results in a particular state, and can be temporally modified in all the
ways that accomplishments can. Clearly the classification of an event is not
solely governed by the verb, but by the semantics of the entire expression in
context.

The second point is that while classifications such as this one are often
useful, they do not *explain* why it is that events expressed in natural lan-
guages fall into these particular classes. We will revisit this issue in Chap-
ter 16 where we will sketch a representational approach due to Dowty (1979)
that accounts for these classes.

Representing Beliefs

There are a fair number of words and expressions that have what might be
called a *world creating* ability. By this, we mean that their meaning repre-
sentations contain logical formulas that are not intended to taken as true in
the real world, but rather as part of some kind of hypothetical world. In addi-
tion, these meaning representations often denote a relation from the speaker,
or some other entity, to this hypothetical world. Examples of words that
have this ability are *believe*, *want*, *imagine* and *know*. World-creating words
generally take various sentence-like constituents as arguments.

Consider the following example:

(14.72) I believe that Mary ate British food.

Applying our event-oriented approach we would say that there two events
underlying this sentence: a believing event relating the speaker to some spe-

cific belief, and an eating event that plays the role of the believed thing.
Ignoring temporal information, a straightforward application of our reified
event approach would produce the following kind of representation:

$$\exists u, v\ ISA(u, Believing) \wedge ISA(v, Eating)$$
$$\wedge Believer(u, Speaker) \wedge BelievedProp(u, v)$$
$$\wedge Eater(v, Mary) \wedge Eaten(v, BritishFood)$$

This seems relatively straightforward, all the right roles are present
and the two events are tied together in a reasonable way. Recall, however,
that in conjunctive representations like this all of the individual conjuncts
must be taken to be true. In this case, this results in a statement that there
actually was an eating of British food by Mary. Specifically, by breaking
this formula apart into separate formulas by conjunction elimination, the
following formula can be produced:

$$\exists v\ ISA(v, Eating)$$
$$\wedge Eater(v, Mary) \wedge Eaten(v, BritishFood)$$

This is clearly more than we want to say. The fact that the speaker believes
this proposition does not make it true; it is only true in the world represented
by the speaker's beliefs. What is needed is a representation that has a struc-
ture similar to this, but where the *Eating* event is given a special status.

Note that reverting to the simpler predicate representations we used
earlier in this chapter does not help. A common mistake using such rep-
resentations would be to represent this sentence with the following kind of
formula:

$$Believing(Speaker, Eating(Mary, BritishFood))$$

The problem with this representation is that it is not even valid FOPC. The
second argument to the *Believing* predicate should be a FOPC term, not a
formula. This syntactic error reflects a deeper semantic problem. Predicates
in FOPC hold between the objects in the domain being modeled, not between
the relations that hold among the objects in the domain. Therefore, FOPC
lacks a meaningful way to assert relations about full propositions, which is
unfortunately exactly what words like *believe*, *want*, *imagine* and *know* want
to do.

The standard method for handling this situation is to augment FOPC
with *operators* that allow us to make statements about full logical formu-
las. Let's consider how this approach might work in the case of example
(14.72). We can introduce an operator called *Believes* that takes two FOPC
formulas as its arguments: a formula designating a believer, and a formula

designating the believed proposition. Applying this operator would result in the following meaning representation:

$$Believes(Speaker, \exists v ISA(v, Eating) \wedge Eater(v, Mary) \wedge Eaten(v, BritishFood))$$

Under this approach, the contribution of the word *believes* to this meaning representation is not a FOPC proposition at all, but rather an operator that is applied to the believed proposition. Therefore, as we discuss in Chapter 15, these world creating verbs play quite a different role in the semantic analysis than more ordinary verbs like *eat*.

As one might expect, keeping track of who believes what about whom at any given point in time gets rather complex. As we will see in Chapter 18, this is an important task in interactive systems that must track users' beliefs as they change during the course of a dialogue.

MODAL OPERATORS
MODAL LOGIC

Operators like *Believes* that apply to logical formulas are known as **modal operators**. Correspondingly, a logic augmented with such operators is known as a **modal logic**. Modal logics have found many uses in the representation of commonsense knowledge in addition to the modeling of belief, among the more prominent are representations of time and hypothetical worlds.

Not surprisingly, modal operators and modal logics raise a host of complex theoretical and practical problems that we cannot even begin to do justice to here. Among the more important issues are the following:

- How inference works in the presence of specific modal operators.
- The kinds of logical formula that particular operators can be applied to.
- How modal operators interact with quantifiers and logical connectives.
- The influence of these operators on the equality of terms across formulas.

The last issue in this list has consequences for modeling agent's knowledge and beliefs in dialogue systems and deserves some elaboration here. In standard FOPC systems, logical terms that are known to be equal to one another can be freely substituted without having any effect on the truth of sentences they occur in. Consider the following examples:

(14.73) Snow has delayed Flight 1045.

(14.74) John's sister's flight serves dinner.

Assuming that these two flights are the same, substituting *Flight 1045* for *John's sister's flight* has no effect on the truth of either sentence.

Now consider, the following variation on the first example:

(14.75) John knows that snow has delayed Flight 1045.

(14.76) John knows that his sister's flight serves dinner.

Here the substitution does not work. John may well know that Flight 1045 has been delayed without knowing that his sister's flight is delayed, simply because he may not know the number of his sister's flight. In other words, even if we assume that these sentences are true, and that John's sister is on Flight 1045, we can not say anything about the truth of the following sentence:

(14.77) John knows that snow has delayed his sister's flight.

Settings like this where a modal operator like *Know* is involved are called **referentially opaque**. In referentially opaque settings, substitution of equal terms may or may not succeed. Ordinary settings where such substitutions always work are said to be **referentially transparent**.

REFERENTIALLY OPAQUE
REFERENTIALLY TRANSPARENT

Pitfalls

As noted in Section 14.3, there are a number of common mistakes in representing the meaning of natural language utterances, that arise from confusing, or equating, elements from real languages with elements in FOPC. Consider the following example, which on the surface looks like a candidate for a standard implication rule:

(14.78) If you're interested in baseball, the Rockies are playing tonight.

A straightforward translation of this sentence into FOPC might look something like this:

$$HaveInterestIn(Hearer, Baseball) \Rightarrow Playing(Rockies, Tonight)$$

This representation is flawed for a large number of reasons. The most obvious ones arise from the semantics of FOPC implications. In the event that the hearer is not interested in baseball, this formula becomes meaningless. Specifically, we can not draw any conclusion about the consequent clause when the antecedent is false. But of course this is a ridiculous conclusion, we know that the Rockies game will go forward regardless of whether or not the hearer happens to like baseball. Exercise 14.10 asks you to come up with a more reasonable FOPC translation of this example.

Now consider the following example:

(14.79) One more beer and I'll fall off this stool.

Again, a simple-minded translation of this sentence might consist of a con-
junction of two clauses: one representing a drinking event and one represent-
ing a falling event. In this case, the surface use of the word *and* obscures the
fact that this sentence instead has an implication underlying it. The lesson
of both of these examples is that English words like *and*, *or* and *if* are only
tenuously related to the elements of FOPC with the same names.

Along the same lines, it is important to remember the complete lack
of significance of the names we make use of in representing FOPC formulas.
Consider the following constant:

InexpensiveVegeta· ‘anIndianFoodOnTuesdays

Despite its impressive morphology, this term, by itself, has no more meaning
than a constant like *X99* would have. See McDermott (1976) for a discourse
on the inherent dangers of such naming schemes.

14.5 Related Representational Approaches

Over the years, a fair number of representational schemes have been invented
to capture the meaning of linguistic utterances for use in natural language
processing systems. Other than logic, two of the most widely used schemes
have been **semantic networks** and **frames**, which are also known as **slot-filler** representations. The KL-ONE (Brachman and Schmolze, 1985), and KRL (Bobrow and Winograd, 1977) systems were influential efforts to represent knowledge for use in natural language processing systems.

SEMANTIC NETWORKS
FRAMES

In semantic networks, objects are represented as nodes in a graph, with relations between objects being represented by named links. In frame-based systems, objects are represented as feature-structures similar to those discussed in Chapter 11, which can, of course, also be naturally represented as graphs. In this approach features are called slots and the values, or fillers, of these slots can either be atomic values or other embedded frames. The following diagram illustrates how example (14.72) might be captured in a frame-based approach.

I believe Mary ate British food.

$$\begin{bmatrix} \text{BELIEVING} & & \\ \text{BELIEVER} & \text{SPEAKER} & \\ \text{BELIEVED} & \begin{bmatrix} \text{EATING} & \\ \text{EATER} & \text{MARY} \\ \text{EATEN} & \text{BRITISHFOOD} \end{bmatrix} \end{bmatrix}$$

now widely accepted that meanings represented in these approaches can
translated into equivalent statements in FOPC with relative ease.

14.6 Alternative Approaches to Meaning

The notion that the translation of linguistic inputs into a formal representation made up of discrete symbols adequately captures the notion of meaning is not surprisingly, subject to a considerable amount of debate. The following sections give brief, wholly inadequate, overviews of some of the major concerns in these debates.

Meaning as Action

An approach that holds considerable appeal when we consider the semantics of imperative sentences is the notion of **meaning as action**. Under this view, utterances are viewed as actions, and the meanings of these utterances resides in **procedures** that are activated in the hearer as a result of hearing the utterance. This approach was followed in the creation of the historically important SHRDLU system, and is summed up well by its creator Terry Winograd (1972b).

MEANING AS ACTION

> One of the basic viewpoints underlying the model is that all language use can be thought of as a way of activating procedures within the hearer. We can think of an utterance as a program—one that indirectly causes a set of operations to be carried out within the hearer's cognitive system.

A recent procedural model of semantics is the **executing schema** or **x-schema** model of Bailey et al. (1997), Narayanan (1997a, 1997b), and Chang et al. (1998). The intuition of this model is that various parts of the semantics of events, including the *aspectual* factors discussed on page 530, are based on schematized descriptions of sensory-motor processes like inception, iteration, enabling, completion, force, and effort. The model represents the

X-SCHEMA

aspectual semantics of events via a kind of probabilistic automaton called a **Petri net** (Murata, 1989). The nets used in the model have states like *process*, *finish*, *suspend*, and *result*.

The meaning representation of an example like *Jack is walking to the store* activates the *process* state of the walking event. An accomplishment event like *Jack walked to the store* activates the *result* state. An iterative activity like *Jack walked to the store every week* is simulated in the model by an iterative activation of the *process* and *result* nodes. This idea of using sensory-motor primitives as a foundation for semantic description is also based on the work of Regier (1996) on the role of visual primitives in a computational model of learning the semantics of spatial prepositions.

Meaning as Truth

The role of formal meaning representations in linguistics, natural language processing, artificial intelligence, and cognitive modeling, is quite different from its role in more philosophical circles. In the former approaches, the name of the game is getting from linguistic inputs to appropriate, unambiguous, and operationally useful representations.[3]

To philosophers, however, the mere translation of a sentence from its original natural form to another artificial form does not get us any closer to its meaning (Lewis, 1972). Formal representations may facilitate real semantic work, but are not by themselves of much interest. Under this view, the important work is in the functions, or procedures, that determine the mapping from these representations to the world being modeled. Of particular interest in these approaches are the functions that determine the **truth conditions** of sentences, or their formal representations.

TRUTH CONDITIONS

14.7 Summary

This chapter has introduced the representational approach to meaning. The following are some of the highlights of this chapter:

- A major approach to meaning in computational linguistics involves the creation of **formal meaning representations** that capture the meaning-related content of linguistic inputs. These representations are intended to bridge the gap from language to commonsense knowledge of the

[3] Of course, what counts as useful varies considerably among these areas.

world.

- The frameworks specify the syntax and semantics of these representations are called **meaning representation languages**. A wide variety of such languages are used in natural language processing and artificial intelligence.
- Such representations need to be able to support the practical computational requirements of semantic processing. Among these are the need to **determine the truth of propositions**, to support **unambiguous representations**, to represent **variables**, to support **inference**, and to be sufficiently **expressive.**
- Human languages have a wide variety of features that are used to convey meaning. Among the most important of these is the ability to convey a **predicate-argument structure**.
- **First Order Predicate Calculus** is a well-understood computationally tractable meaning representation language that offers much of what is needed in a meaning representation language.
- Important classes of meaning including bf categories, **events**, and **time** can be captured in FOPC. Propositions corresponding to such concepts as **beliefs** and **desires** require extensions to FOPC including **modal operators**.
- **Semantic networks** and **frames** can be captured within the FOPC framework.

Bibliographical and Historical Notes

The earliest computational use of declarative meaning representations in natural language processing was in the context of question-answering systems (Green et al., 1961; Raphael, 1968; Lindsey, 1963). These systems employed ad-hoc representations for the facts needed to answer questions. Questions were then translated into a form that could be matched against facts in the knowledge base. Simmons (1965) provides an overview of these early efforts.

Woods (1967) investigated the use of FOPC-like representations in question answering as a replacement for the ad-hoc representations in use at the time. Woods (1973) further developed and extended these ideas in the landmark Lunar system. Interestingly, the representations used in Lunar had

both a truth-conditional and a procedural semantics. Winograd (1972b) em- ployed a similar representation based on the Micro-Planner language in his SHRDLU system.

During this same period, researchers interested in the cognitive model- ing of language and memory had been working with various forms of asso- ciative network representations. Masterman (1957) was probably the first to make computational use of a semantic network-like knowledge representa- tion, although semantic networks are generally credited to Quillian (1968). A considerable amount work in the semantic network framework was car- ried out during this era (Norman and Rumelhart, 1975; Schank, 1972; Wilks, 1975c, 1975b; Kintsch, 1974). It was during this period that a number of researchers began to incorporate Fillmore's notion of case roles (Fillmore, 1968) into their representations. Simmons (1973) was the earliest adopter of case roles as part of representations for natural language processing.

Detailed analyses by Woods (1975) and Brachman (1979) aimed at fig- uring out what semantic networks actually mean led to the development of a number of more sophisticated network-like languages including KRL (Bo- brow and Winograd, 1977) and KL-ONE (Brachman and Schmolze, 1985). As these frameworks became more sophisticated and well-defined it became clear that they were restricted variants of FOPC coupled with specialized in- ference procedures. A useful collection of papers covering much of this work can be found in (Brachman and Levesque, 1985). Russell and Norvig (1995) describe a modern perspective on these representational efforts.

Linguistic efforts to assign semantic structures to natural language sen- tences in the generative era began with the work of Katz and Fodor (1963). The limitations of their simple feature-based representations and the natu- ral fit of logic to many of linguistic problems of the day quickly led to the adoption of a variety of predicate-argument structures as preferred semantic representations (Lakoff, 1972; McCawley, 1968). The subsequent introduc- tion by Montague (1973) of truth-conditional model-theoretic framework into linguistic theory led to a much tighter integration between theories of formal syntax and a wide range of formal semantic frameworks. Good in- troductions to Montague semantics and its role in linguistic theory can be found in (Dowty et al., 1981; Partee, 1976).

The representation of events as reified objects is due to Davidson (1967). The approach presented here, which explicitly reifies event participants, is due to Parsons (1990). The use of modal operators and in the representation of knowledge and belief is due to Hintikka (1969). Moore (1977) was the first to make computational use of this approach. Fauconnier (1985) deals

...a wide range of issues relating to beliefs and belief spaces from a cogni- ...science perspective. Most current computational approaches to temporal ...soning are based on Allen's notion of temporal intervals (Allen, 1984). ...Meulen (1995) provides a modern treatment of tense and aspect. Davis ...990) describes the use of FOPC to represent knowledge across a wide range ...common sense domains including quantities, space, time, and beliefs.

A recent comprehensive treatment of logic and language can be found ...(van Benthem and ter Meulen, 1997). The classic semantics text is (Lyons, ...977). McCawley (1993) is an indispensable textbook covering a wide range ...of topics concerning logic and language. Chierchia and McConnell-Ginet ...991) also provides broad coverage of semantic issues from a linguistic ...perspective. Heim and Kratzer (1998) is a more recent text written from the ...perspective of current generative theory.

EXERCISES

14.1 Choose a recipe from your favorite cookbook and try to make explicit all the common-sense knowledge that would be needed to follow it.

14.2 Proponents of information retrieval occasionally claim that natural language texts in their raw form are a perfectly suitable source of knowledge for question answering. Sketch an argument against this claim.

14.3 Peruse your daily newspaper for three examples of ambiguous sentences. Describe the various sources of the ambiguities.

14.4 Consider a domain where the word *coffee* can refer to the following concepts in a knowledge-based: a caffeinated or decaffeinated beverage, ground coffee used to make either kind of beverage, and the beans themselves. Give arguments as to which of the following uses of coffee are ambiguous and which are vague.

a. I've had my coffee for today.

b. Buy some coffee on your way home.

c. Please grind some more coffee.

14.5 Encode in FOPC as much of the knowledge as you can that you came up with for Exercise 14.1

14.6 The following rule, which we gave as a translation for Example 14.4, is not a reasonable definition of what it means to be a vegetarian restaurant.

$$\forall x \, VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$$

Give a FOPC rule that better defines vegetarian restaurants in terms of what they serve.

14.7 Give a FOPC translations for the following sentences:

a. Vegetarians do not eat meat.

b. Not all vegetarians eat eggs.

14.8 Give a set of facts and inferences necessary to prove the following assertions:

a. McDonalds is not a vegetarian restaurant.

b. Some vegetarians can eat at McDonalds.

Don't just place these facts in your knowledge base. Show that they can be inferred from some more general facts about vegetarians and McDonalds.

14.9 Give FOPC translations for the following sentences that capture the temporal relationships between the events.

a. When Mary's flight departed, I ate lunch.

b. When Mary's flight departed, I had eaten lunch.

14.10 Give a reasonable FOPC translation of the following example.

If you're interested in baseball, the Rockies are playing tonight.

14.11 On Page 516 we gave the following FOPC translation for Example 14.17.

$$Have(Speaker, FiveDollars) \wedge \neg Have(Speaker, LotOfTime)$$

This literal representation would not be particularly useful to a restaurant-oriented question answering system. Give a deeper FOPC meaning representation for this example that is closer to what it really means.

14.12 On Page 516, we gave the following representation as a translation for the sentence *Ay Caramba is near ICSI.*

$$Near(LocationOf(AyCaramba), LocationOf(ICSI))$$

In our truth-conditional semantics, this formula is either true or false given the contents of some knowledge-base. Critique this truth-conditional approach with respect to the meaning of words like *near*.

15 SEMANTIC ANALYSIS

"Then you should say what you mean," the March Hare went on.
"I do," Alice hastily replied; "at least–at least I mean what I say–that's the same thing, you know."
"Not the same thing a bit!" said the Hatter. "You might just as well say that 'I see what I eat' is the same thing as 'I eat what I see'!"

Lewis Carroll, *Alice in Wonderland*

This chapter presents a number of computational approaches to the problem of **semantic analysis**, the process whereby meaning representations of the kind discussed in the previous chapter are composed and assigned to linguistic inputs. As we will see in this and later chapters, the creation of rich and accurate meaning representations necessarily involves a wide range of knowledge-sources and inference techniques. Among the sources of knowledge that are typically used are the meanings of words, the meanings associated with grammatical structures, knowledge about the structure of the discourse, knowledge about the context in which the discourse is occurring, and common-sense knowledge about the topic at hand.

SEMANTIC ANALYSIS

The first approach we cover is a kind of **syntax-driven semantic analysis** that is fairly limited in its scope. It assigns meaning representations to inputs based solely on static knowledge from the lexicon and the grammar. In this approach, when we refer to an input's meaning, or meaning representation, we have in mind an impoverished representation that is both context independent and inference free. Meaning representations of this type correspond to the notion of a literal meaning introduced in the last chapter.

SYNTAX-DRIVEN SEMANTIC ANALYSIS

PEARSON EDUCATION SERIES IN ARTIFICIAL INTELLIGENCE
Stuart Russell and Peter Norvig, Editors

GRAHAM — ANSI *Common Lisp*
RUSSELL & NORVIG — *Artificial Intelligence: A Modern Approach*
JURAFSKY & MARTIN — *Speech and Language Processing*

Speech and Language Processing

An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition

Daniel Jurafsky and James H. Martin

University of Colorado, Boulder

Contributing writers:
Andrew Kehler, Keith Vander Linden, and Nigel Ward

PEARSON Education

ISBN 81-7808-594-1

First Indian Reprint, 2002
Second Indian Reprint, 2003
Third Indian Reprint, 2004

This edition is manufactured in India and is authorized for sale only in India, Bangladesh, Pakistan, Nepal, Sri Lanka and the Maldives.

Published by Pearson Education (Singapore) Pte. Ltd., Indian Branch, 482 F.I.E. Patparganj, Delhi 110 092, India

Printed in India by Saurabh Printers Pvt. Ltd.

For my parents, Ruth and Al Jurafsky — D.J.

For Linda — J.M.