

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
REPRESENTA-
TIONS

MEANING
REPRESENTA-
TION
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:

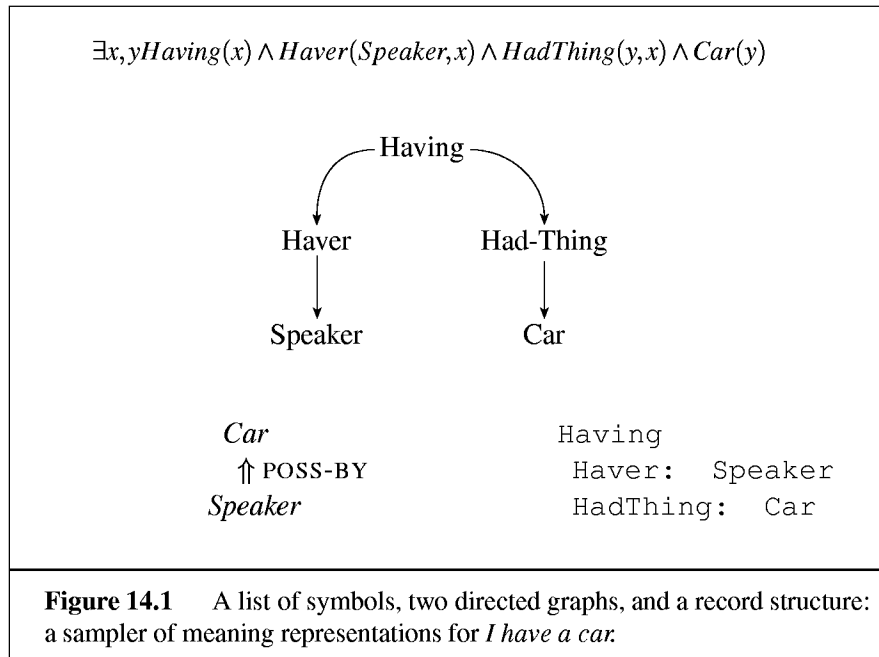
- Answering and grading essay questions requires background knowledge about the topic of the question, the desired knowledge level of the students, and how such questions are *normally* answered.
- Reading a menu and deciding what to order, giving advice about where to go to dinner, following a recipe, 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 common-sense 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.

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

LITERAL
MEANING

representing what is sometimes called the **literal meaning** of sentences. By this, we have in mind representations that are closely tied to the conventional 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 representations 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 discussion, 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 representation must fulfill, and some of the complications that inevitably arise in the process of designing such meaning representations. In each of these examples, 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 representation: it must be possible to use the representation to determine the relationship 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 about its world.

KNOWLEDGE
BASE

In this example, let us assume that the meaning of this question contains, 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 knowledge base of facts about a set of restaurants. If the system finds a representation matching the input proposition in its knowledge base, it can return an affirmative answer. Otherwise, it must either say *No*, if its knowledge of local restaurants is complete, or say that it does not know if there is reason to believe that its knowledge is incomplete.

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

VERIFIABILITY

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 circumstances 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.

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

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 correct interpretations. Since we reason about, and act upon, the semantic content of linguistic inputs, the final representation of an input's meaning should be free from any ambiguity. Therefore, regardless of any ambiguity in the raw input, it is critical that a meaning representation language support 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?

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.

Given that these alternatives use different words and have widely varying syntactic analyses, it would not be unreasonable to expect them to have substantially different meaning representations. Such a situation would, however, have undesirable consequences for our matching approach to determining the truth of our representations. If the system's knowledge base contains only a single representation of the fact in question, then the representations underlying all but one of our alternatives will fail to produce a match. We could, of course, store all possible alternative representations of the same fact in the knowledge base, but this would lead to an enormous number 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 containing these words.

WORD
SENSES

WORD SENSE
DISAMBIGUA-
TION

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 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 con-

nection 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.

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.

INFERENCE

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 lan-

guage 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 Chapter 9 and Chapter 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 *NPs*.
- 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.

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

THEMATIC
ROLE
CASE ROLE

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 **selection restriction**. Through the use of these selection restrictions, verbs can specify semantic restrictions on their arguments.

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, this support 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.

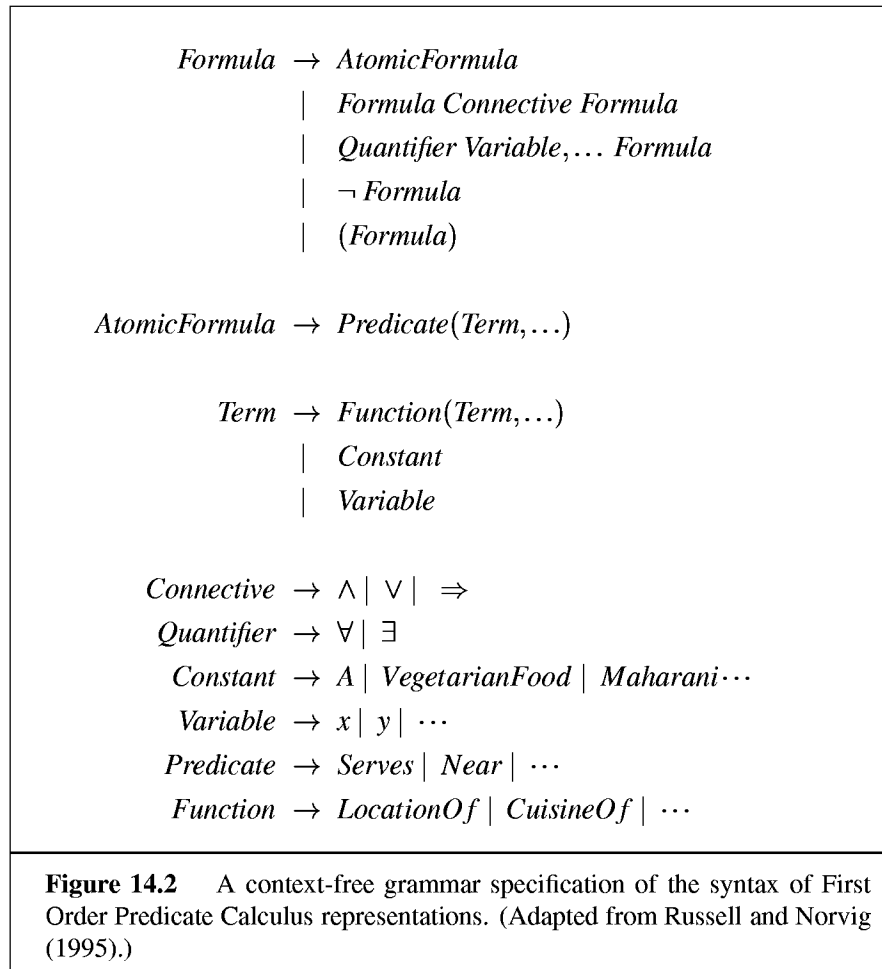
Constants in FOPC refer to specific objects in the world being described. Such constants are conventionally depicted as either single capitalized letters such as *A* and *B* or single capitalized words that are often reminiscent of proper nouns such as *Maharani* and *Harry*. Like programming language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

Functions in FOPC correspond to concepts that which are often expressed in English as genitives such as *the location of Maharani* or *Maharani's location*. A FOPC translation of such an expression might look like

TERM

CONSTANTS

FUNCTIONS



the following.

LocationOf(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.

Serves(Maharani, 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*.

Restaurant(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

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	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>False</i>	<i>True</i>
<i>False</i>	<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>
<i>True</i>	<i>False</i>	<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>
<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>True</i>

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.

$$\begin{aligned} \exists x & Restaurant(x) \\ & \wedge Serves(x, MexicanFood) \\ & \wedge Near((LocationOf(x), LocationOf(ICS))) \end{aligned}$$

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.

$$\begin{aligned} & Restaurant(AyCaramba) \\ & \wedge Serves(AyCaramba, MexicanFood) \\ & \wedge Near((LocationOf(AyCaramba), LocationOf(ICS))) \end{aligned}$$

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.

$$\begin{aligned} & \text{VegetarianRestaurant}(\text{Maharani}) \\ & \Rightarrow \text{Serves}(\text{Maharani}, \text{VegetarianFood}) \end{aligned}$$

If we assume that we know that the consequent clause,

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

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 Table 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 objects that are not vegetarian restaurants? Consider the substitution of a non-vegetarian restaurant such as *Ay Caramba's* for the variable x .

$$\begin{aligned} & \text{VegetarianRestaurant}(\text{AyCaramba}) \\ & \Rightarrow \text{Serves}(\text{AyCaramba}, \text{VegetarianFood}) \end{aligned}$$

Since the antecedent of the implication is *False*, we can determine from Table 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.

$$\begin{aligned} & \text{VegetarianRestaurant}(\text{Carburetor}) \\ & \Rightarrow \text{Serves}(\text{Carburetor}, \text{VegetarianFood}) \end{aligned}$$

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.

(14.21)

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

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

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

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

PRODUCTION
SYSTEMS

In **backward chaining**, modus ponens is run in reverse to prove specific 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.

BACKWARD
CHAINING

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 for 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,
RESOLUTION	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.

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 selection 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-fledged objects. It is, therefore, difficult to make assertions about categories themselves, rather than about their individual members. For example, we might want to designate the most popular member of a given category as in the following expression.

MostPopular(Maharani, VegetarianRestaurant)

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

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

REIFICATION

ISA(Maharani, VegetarianRestaurant)

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

AKO(VegetarianRestaurant, Restaurant)

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

Chapter 16 discusses the practical use of such relations in databases of lexical relations, in the representation of selection 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 representa-

tions 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

$$\forall w, x, y, z \text{ } Eating_7(w, x, y, z) \Rightarrow Eating_6(w, x, y)$$

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.

$$\begin{aligned} &\exists w, x, y \text{ Eating}(\text{Speaker}, w, x, y) \\ &\exists w, x \text{ Eating}(\text{Speaker}, \text{TurkeySandwich}, w, x) \\ &\exists w \text{ Eating}(\text{Speaker}, \text{TurkeySandwich}, w, \text{Desk}) \\ &\exists w, x \text{ Eating}(\text{Speaker}, w, x, \text{Desk}) \\ &\exists w, x \text{ Eating}(\text{Speaker}, w, \text{Lunch}, x) \\ &\exists w \text{ Eating}(\text{Speaker}, \text{TurkeySandwich}, \text{Lunch}, w) \\ &\text{Eating}(\text{Speaker}, \text{TurkeySandwich}, \text{Lunch}, \text{Desk}) \end{aligned}$$

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 (ie. 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.

$$\begin{aligned} &\exists w, x \text{ Eating}(\text{Speaker}, w, x, \text{Desk}) \\ &\exists w, x \text{ Eating}(\text{Speaker}, w, \text{Lunch}, x) \\ &\exists w, x \text{ Eating}(\text{Speaker}, w, \text{Lunch}, \text{Desk}) \end{aligned}$$

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.

$$\begin{aligned} &\exists w \text{ ISA}(w, \text{Eating}) \\ &\quad \wedge \text{Eater}(w, \text{Speaker}) \wedge \text{Eaten}(w, \text{TurkeySandwich}) \end{aligned}$$

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.

$$\begin{aligned} &\exists w \text{ ISA}(w, \text{Eating}) \wedge \text{Eater}(w, \text{Speaker}) \\ &\exists w \text{ ISA}(w, \text{Eating}) \\ &\quad \wedge \text{Eater}(w, \text{Speaker}) \wedge \text{Eaten}(w, \text{TurkeySandwich}) \\ &\quad \wedge \text{MealEaten}(w, \text{Lunch}) \end{aligned}$$

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 been 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.

$$\begin{aligned} \exists w \text{ ISA}(w, \text{Arriving}) \\ \wedge \text{Arriver}(w, \text{Speaker}) \wedge \text{Destination}(w, \text{NewYork}) \end{aligned}$$

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.

$$\begin{aligned} \exists i, e, w, t \text{ ISA}(w, \text{Arriving}) \\ \wedge \text{Arriver}(w, \text{Speaker}) \wedge \text{Destination}(w, \text{NewYork}) \\ \text{IntervalOf}(w, i) \wedge \text{EndPoint}(i, e) \wedge \text{Precedes}(e, \text{Now}) \end{aligned}$$

$$\begin{aligned} \exists i, e, w, t \text{ ISA}(w, \text{Arriving}) \\ \wedge \text{Arriver}(w, \text{Speaker}) \wedge \text{Destination}(w, \text{NewYork}) \\ \text{IntervalOf}(w, i) \wedge \text{MemberOf}(i, \text{Now}) \end{aligned}$$

$$\begin{aligned} \exists i, e, w, t \text{ ISA}(w, \text{Arriving}) \\ \wedge \text{Arriver}(w, \text{Speaker}) \wedge \text{Destination}(w, \text{NewYork}) \\ \text{IntervalOf}(w, i) \wedge \text{EndPoint}(i, e) \wedge \text{Precedes}(\text{Now}, e) \end{aligned}$$

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 (eg. 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.

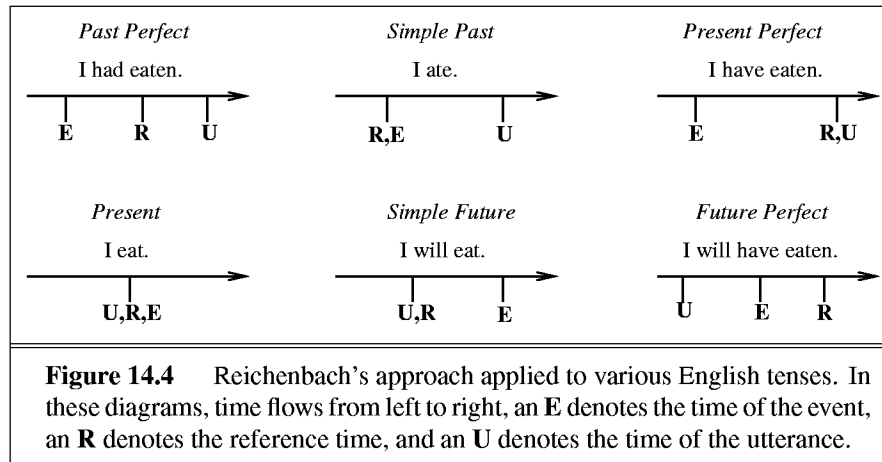
REFERENCE
POINT

(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, ie. 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,



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, 1983a). 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

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

(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

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

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.

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

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

(14.62) *I lived in Brooklyn in a year.

(14.63) She booked a flight in a minute.

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

The final aspectual class, **achievements**, are similar to accomplishments in that they result in a state. Consider the following examples.

ACHIEVEMENTS

(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 an achievement example is unacceptable.

TELIC EVEN-
TUALITIES

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 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 languages fall into these particular classes. We will revisit this issue in Chapter 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 representations contain logical formulas that are not intended to be taken as true in the real world, but rather as part of some kind of hypothetical world. In addition, 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 are 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.

$$\begin{aligned} \exists u, v \text{ ISA}(u, \text{Believing}) \wedge \text{ISA}(v, \text{Eating}) \\ \wedge \text{Believer}(u, \text{Speaker}) \wedge \text{BelievedProp}(u, v) \\ \wedge \text{Eater}(v, \text{Mary}) \wedge \text{Eaten}(v, \text{BritishFood}) \end{aligned}$$

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.

$$\begin{aligned} \exists v \text{ ISA}(v, \text{Eating}) \\ \wedge \text{Eater}(v, \text{Mary}) \wedge \text{Eaten}(v, \text{BritishFood}) \end{aligned}$$

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 structure 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 representations would be to represent this sentence with the following kind of formula.

$$\text{Believing}(\text{Speaker}, \text{Eating}(\text{Mary}, \text{BritishFood}))$$

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 formulas. 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.

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

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 dialog.

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 can not 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 dialog 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**.

REFEREN-
TIALY
OPAQUE

REFEREN-
TIALY
TRANSPAR-
ENT

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 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.

$$\begin{aligned} &HaveInterestIn(Hearer, Baseball) \\ &\Rightarrow Playing(Rockies, Tonight) \end{aligned}$$

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 conjunction of two clauses: one representing a drinking event and one representing 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.

InexpensiveVegetarianIndianFoodOnTuesdays

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.

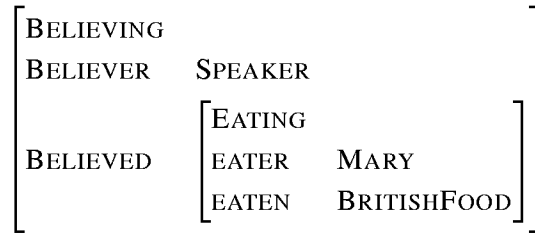
14.5 RELATED REPRESENTATIONAL APPROACHES

SEMANTIC
NETWORKS
FRAMES

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, 1985a), and KRL (Bobrow and Winograd, 1977) systems represent influential efforts to represent knowledge for use in natural language processing systems.

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.



It is now widely accepted that meanings represented in these approaches can be 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 526, are based on schematized descriptions of sensory-motor processes like inception, iteration, enabling, completion, force, and effort. The model represents

X-SCHEMA

the 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 *ready*, *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 representation, to represent variables, to support inference, and to be 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.
- FOPC 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 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.*, 1963; 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) employed a similar representation based on the Micro-Planner language in his SHRDLU system.

During this same period, researchers interested in the cognitive modeling of language and memory had been working with various forms of associative network representations. Masterman (1957) was probably the first to make computational use of a semantic network-like knowledge representation, although semantic networks are generally credited to Quillian (1968). A considerable amount of work in the semantic network framework was carried 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 (1973a) was the earliest adopter of case roles as part of representations for natural language processing.

Detailed analyses by Woods (1975) and Brachman and Schmolze (1985a) aimed at figuring out what semantic networks actually mean led to the development of a number of more sophisticated network-like languages including KRL (Bobrow and Winograd, 1977) and KL-ONE (Brachman and Schmolze, 1985a). As these frameworks became more sophisticated and well-defined it became clear that they were restricted variants of FOPC coupled with specialized inference 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 sentences in the generative era began with the work of Katz and Fodor (1963). The limitations of their simple feature-based representations and the natural 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 introduction 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 introductions 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 modal logic in the representation of knowledge and belief is due to Hintikka (1969a). Moore

(1977) was the first to make computational use of this approach. Fauconnier (1985) deals with a wide range of issues relating to beliefs and belief spaces from a cognitive science perspective. Most current computational approaches to temporal reasoning are based on Allen's notion of temporal intervals (Allen, 1984). ter Meulen (1995) provides a modern treatment of tense and aspect. Davis (1990) describes the use of FOPC to represent knowledge across a wide range of common sense domains including quantities, space, time, and beliefs.

A recent comprehensive treatment of logic and language can be found in (van Benthem and ter Meulen, 1997). The classic semantics text is (Lyons, 1977). McCawley (1993) is an indispensable textbook covering a wide range of topics concerning logic and language. Chierchia and McConnell-Ginet (1991) 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-base: 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 14.20, is not a reasonable definition of what it means to be a vegetarian restaurant.

$$\forall x \text{VegetarianRestaurant}(x) \Rightarrow \text{Serves}(x, \text{VegetarianFood})$$

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 512 we gave the following FOPC translation for Example 14.17.

$$\text{Have}(\text{Speaker}, \text{FiveDollars}) \wedge \neg \text{Have}(\text{Speaker}, \text{LotOfTime})$$

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 Describe, in English, the knowledge that would be needed to infer the deeper representation you produced for the last exercise from the initial literal representation.

14.13 On Page 512, 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.

There are two reasons for proceeding along these lines: there are some limited application domains where such representations are sufficient to pro-

duce useful results, and these impoverished representations can serve as inputs to subsequent processes that can produce richer, more useful, meaning representations. Chapters 18 and 19 will show how these meaning representations can be used in processing extended discourses, while Chapter 21 will show how they can be used in machine translation.

Section 15.5 then presents two alternative approaches to semantic analysis that are more well-suited to practical applications. The first approach, **semantic grammars**, has been widely applied in the construction of interactive dialog systems. In this approach, the elements of the grammars are strongly motivated by the semantic entities and relations of the domain being discussed. As we will see, the actual algorithms used in this approach are quite similar to those described in Section 15.1. The difference lies in the grammars that are used.

The final approach, presented in Section 15.5, addresses the task of extracting small amounts of pertinent information from large bodies of text. As we will see, this **information extraction** task does not require the kind of complete syntactic analysis assumed in the other approaches. Instead, a series of quite limited, mostly finite-state, automata are combined via a **cascade** to produce a robust semantic analyzer.

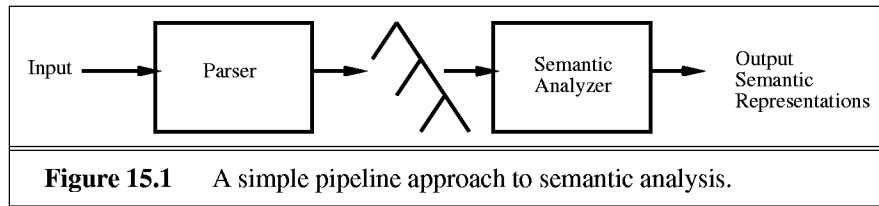
15.1 SYNTAX-DRIVEN SEMANTIC ANALYSIS

PRINCIPLE OF
COMPOSITIONALITY

The approach detailed in this section is based on the **principle of compositionality**.¹ The key idea underlying this approach is that the meaning of a sentence can be composed from the meanings of its parts. Of course, when interpreted superficially, this principle is somewhat less than useful. We know that sentences are composed of words, and that words are the primary carriers of meaning in language. It would seem then that all this principle tells us is that we should compose the meaning representation for sentences from the meanings of the words that make them up.

Fortunately, the Mad Hatter has provided us with a hint as to how to make this principle useful. The meaning of a sentence is not based solely on the words that make it up, it is based on the ordering, grouping, and relations among the words in the sentence. Of course, this is simply another way

¹ This is normally referred to as Frege's principle of compositionality. There appears to be little reason for this ascription, since the principle never explicitly appears in any of his writings. Indeed, many of his writings can be taken as supporting a decidedly non-compositional view. Janssen (1997) discusses this topic in more detail.



of saying that the meaning of a sentence is partially based on its syntactic structure. Therefore, in *syntax-driven semantic analysis*, the composition of meaning representations is guided by the syntactic *components* and *relations* provided by the kind of grammars discussed in Chapters 9, 11, and 12.

We can begin by assuming that the syntactic analysis of an input sentence will form the input to a semantic analyzer. Figure 15.1 illustrates the obvious pipeline-oriented approach that follows directly from this assumption. An input is first passed through a parser to derive its syntactic analysis. This analysis is then passed as input to a **semantic analyzer** to produce a meaning representation. Note that although this diagram shows a parse tree as input, other syntactic representations such as feature structures, or lexical dependency diagrams, can be used. The remainder of this section will assume tree-like inputs.

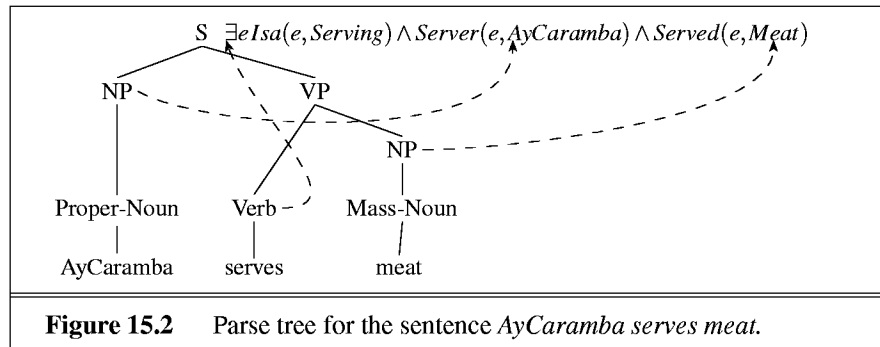
SEMANTIC
ANALYZER

Before moving on, we should make explicit a major assumption about the role ambiguity of this approach. In the syntax driven approach presented here, ambiguities arising from the syntax and the lexicon will lead to the creation of multiple ambiguous meaning representations. It is not the job of the semantic analyzer, narrowly defined, to resolve these ambiguities. Instead, it is the job of subsequent interpretation processes with access to domain specific knowledge, and knowledge of context to *select* among competing representations. Of course, we can cut down on the number of ambiguous representations produced, through the use of robust part-of-speech taggers, prepositional phrase attachment mechanisms, and, as we will see in Chapter 16, word-sense disambiguation mechanisms.

Let's consider how such an analysis might proceed with the following example.

(15.1) AyCaramba serves meat.

Figure 15.2 shows the simplified parse tree (lacking feature attachments), along with an appropriate meaning representation for this example. As suggested by the dashed arrows, a semantic analyzer given this tree as input might fruitfully proceed by first retrieving a meaning representation from the subtree corresponding to the verb *serves*. The analyzer might next retrieve



meaning representations corresponding to the two noun phrases in the sentence. Then using the representation acquired from the verb as a template, the noun phrase meaning representations can be used to bind the appropriate variables in the verb representation, thus producing the meaning representation for the sentence as a whole.

Unfortunately, there is a rather obvious problem with this simplified story. As described, the function used to interpret the tree in Figure 15.2 must know, among other things, that it is the verb that carries the template upon which the final representation is based, where this verb occurs in the tree, where its corresponding arguments are, and which argument fills which role in the verb's meaning representation. In other words, it requires a good deal of specific knowledge about *this particular example and its parse tree* to create the required meaning representation. Given that there are an infinite number of such trees for any reasonable grammar, any approach based on one semantic function for every possible tree is in serious trouble.

Fortunately, we have faced this problem before. Languages are not defined by enumerating the strings or trees that are permitted, but rather by specifying finite devices that are capable of generating the required set of outputs. It would seem, therefore, that the right place for semantic knowledge in a syntax-directed approach is with the finite set of devices that are used to generate trees in the first place: the grammar rules and the lexical entries. This is known as the **rule to rule hypothesis** (Bach, 1976).

Designing an analyzer based on this approach brings us back to the notion of parts and what it means for them to have meanings. The remainder of this section can be seen as an attempt to answer the following two questions.

- What does it mean for syntactic constituents to have meanings?
- What do these meanings have to be like so that they can be composed into larger meanings?

Semantic Augmentations to Context-Free Grammar Rules

In keeping with the approach begun in Chapter 11, we will begin by augmenting context-free grammar rules with **semantic attachments**. These attachments can be thought of as instructions that specify how to compute the meaning representation of a construction from the meanings of its constituent parts. Abstractly, our augmented rules have the following structure.

SEMANTIC ATTACHMENTS

$$A \rightarrow \alpha_1 \dots \alpha_n \quad \{f(\alpha_j.sem, \dots, \alpha_k.sem)\}$$

The semantic attachment to the basic context-free rule is shown in the $\{\dots\}$ to the right of the rule's syntactic constituents. This notation states that the meaning representation assigned to the construction A , which we will denote as $A.sem$, can be computed by running the function f on some subset of the semantic attachments of A 's constituents.

This characterization of our semantic attachments as a simple function application is rather abstract. To make this notion more concrete, we will walk through the semantic attachments necessary to compute the meaning representation for a series of examples beginning with Example 15.1, shown earlier in Figure 15.2. We will begin with the more concrete entities in this example, as specified by the noun phrases, and work our way up to the more complex expressions representing the meaning of the entire sentence. The concrete entities in this example are represented by the FOPC constants *AyCaramba* and *Meat*. Our first task is to associate these constants with the constituents of the tree that introduce them. The first step toward accomplishing this is to pair them with the lexical rules representing the words that introduce them into the sentence.

$$ProperNoun \rightarrow AyCaramba \quad \{AyCaramba\}$$

$$MassNoun \rightarrow meat \quad \{Meat\}$$

These two rules specify that the meanings associated with the subtrees generated by these rules consist of the constants *AyCaramba* and *Meat*.

Note, however, that as the arrows in Figure 15.2 indicate, the subtrees corresponding to these rules do not directly contribute these FOPC constants to the final meaning representation. Rather, it is the *NPs* higher in the tree that contribute them to the final representation. In keeping with the principle of compositionality, we can deal with this indirect contribution by stipulating that the upper *NPs* obtain their meaning representations from the meanings of their children. In these two cases, we will assume that the meaning representations of the children are simply copied upward to the parents.

$$NP \rightarrow ProperNoun \quad \{ProperNoun.sem\}$$

$$NP \rightarrow MassNoun \quad \{MassNoun.sem\}$$

These rules state that the meaning representation of the noun phrases are the same as the meaning representations of their individual components, denoted by *ProperNoun.sem* and *MassNoun.sem*. In general, it will be the case that for non-branching grammar rules, the semantic expression associated with the child will be copied unchanged to the parent.

Before proceeding, we should point out that there is at least one potentially confusing aspect to this discussion. While the static semantic attachment to our first *NP* rule is simply *ProperNoun.sem*, the semantic value of the tree produced by that rule in this example is *AyCaramba*. It is critical to distinguish between the semantic attachment of a rule, and the semantic value associated with a tree generated by a rule. The first is a set of instructions on how to construct a meaning representation, while the second consists of the result of following those instructions.

Returning to our example, having accounted for the constants in the representation, we can move on to the event underlying this utterance as specified by *serves*. As illustrated in Figure 15.2, a generic *Serving* event involves a *Server* and something *Served*, as captured in the following logical formula.

$$\exists e, x, y \text{ Isa}(e, \text{Serving}) \wedge \text{Server}(e, x) \wedge \text{Served}(e, y)$$

As a first attempt at this verb's semantic attachment, we can simply take this logical formula as *serve*'s semantic attachment, as in the following.

$$\begin{aligned} \text{Verb} &\rightarrow \text{serves} \\ &\{\exists e, x, y \text{ Isa}(e, \text{Serving}) \wedge \text{Server}(e, x) \wedge \text{Served}(e, y)\} \end{aligned}$$

Moving up the parse tree, the next constituent to be considered is the *VP* that dominates both *serves* and *meat*. Unlike the *NPs*, we can not simply copy the meaning of these children up to the parent *VP*. Rather, we need to *incorporate* the meaning of the *NP* into the meaning of the *Verb* and assign the resulting representation to the *VP.sem*. In this case, this consists of replacing the variable *y* with the logical term *Meat* as the second argument of the *Served* role of the *Serves* event. This yields the following meaning representation, which can be glossed as something like *someone serves meat*.

$$\exists e, x \text{ Isa}(e, \text{Serving}) \wedge \text{Server}(e, x) \wedge \text{Served}(e, \text{Meat})$$

To come up with this representation, the semantic attachment for the *VP* must provide a means to replace the quantified variable *y* within the body of *V.sem* with the logical constant *Meat*, as stipulated by *NP.sem*. Abstracting away from this specific example, the *VP* semantic attachment must have two

capabilities: the means to *know* exactly which variables within the *Verb*'s semantic attachment are to be replaced by the semantics of the *Verb*'s arguments, and the ability to perform such a replacement.

Unfortunately, there is no straightforward way to do this given the mechanisms we now have at our disposal. The FOPC formula we attached to the *V.sem* does not provide any advice about when and how each of its three quantified variables should be replaced, and we have no simple way, within our current specification of FOPC, for performing such a replacement even if we did know.

Fortunately, there is a notational extension to FOPC called the **lambda notation** (Church, 1940) that provides exactly the kind of formal parameter functionality that we need. This notation extends the syntax of FOPC to include expressions of the following form.

LAMBDA
NOTATION

$$\lambda x P(x)$$

Such expressions consist of the Greek symbol λ , followed by one or more variables, followed by a FOPC expression that makes use of those variables.

The usefulness of these λ -expressions is based on the ability to *apply* them to logical terms to yield new FOPC expressions where the formal parameter variables are bound to the specified terms. This process is known as λ -reduction and is little more than a simple textual replacement of the λ variables with the specified FOPC terms, accompanied by the subsequent removal of the λ . The following expressions illustrate the application of a λ -expression to the constant *A*, followed by the result of performing a λ -reduction on this expression.

$$\lambda x P(x)(A)$$

$$P(A)$$

This λ -notation provides both of the capabilities we said were needed in the *Verb* semantics: the formal parameter list makes a set of variables within the body available, and the λ -reduction process implements the desired replacement of variables with terms.

An important and useful variation of this technique is the use of one λ -expression as the body of another as in the following expression.

$$\lambda x \lambda y \text{Near}(x, y)$$

This fairly abstract expression can be glossed as the state of something being near something else. The following expressions illustrate a single λ -application and subsequent reduction with this kind of embedded λ -

expression.

$$\lambda x \lambda y \text{Near}(x, y)(ICSI)$$

$$\lambda y \text{Near}(ICSI, y)$$

The important point here is that the resulting expression is still a λ -expression; the first reduction bound the variable x and removed the outer λ , thus revealing the inner expression. As might be expected, this resulting λ -expression can, in turn, be applied to another term to arrive at a fully specified logical formula, as in the following.

$$\lambda y \text{Near}(ICSI, y)(AyCaramba)$$

$$\text{Near}(ICSI, AyCaramba)$$

CURRYING

This technique, called **currying**²(Schönkinkel, 1924), is a way of converting a predicate with multiple arguments into a sequence of single argument predicates. As we will see shortly, this technique is quite useful when the arguments to a predicate do not all appear together as daughters of the predicated in a parse tree.

With the λ -notation and the process of λ -reduction, we have the tools needed to return to the semantic attachments for our *VP* constituent. Recall that what was needed was a way to replace the variable representing the *Served* role with the meaning representation provided by the *NP* constituent of the *VP*. This can be accomplished in two steps: changing the semantic attachment of the *Verb* to a λ -expression, and having the semantic attachment of the *VP* apply this expression to the *NP* semantics. The first of these steps can be accomplished by designating x , the variable corresponding to the *Served* role, as the λ -variable for a λ -expression provided as the semantic attachment for *serve*.

$$\text{Verb} \rightarrow \text{serves}$$

$$\{\lambda x \exists e, y \text{Isa}(e, \text{Serving}) \wedge \text{Server}(e, y) \wedge \text{Served}(e, x)\}$$

This attachment makes the variable x externally available to be bound by an application of this expression to a logical term. The attachment for our transitive *VP* rule, therefore, specifies a λ -application where the λ -expression is provided by *Verb.sem* and the argument is provided by *NP.sem*.

$$\text{VP} \rightarrow \text{Verb NP} \quad \{\text{Verb.sem}(\text{NP.sem})\}$$

This λ -application results in the replacement, or binding, of x , the single formal parameter of the λ -expression, with the value contained in

² *Currying* is the standard term, although Heim and Kratzer (1998) present an interesting argument for the term *Schönkinkelization* over currying, since Curry *later* built on Schönkinkel's work.

NP.sem. A λ -reduction removes the λ revealing the inner expression with the parameter x replaced by the constant *Meat*. This expression, the meaning of the verb phrase *serves meat*, is then the value of *VP.sem*.

$$\exists e, y \text{ Isa}(e, \text{Serving}) \wedge \text{Server}(e, y) \wedge \text{Served}(e, \text{Meat})$$

To complete this example, we must create the semantic attachment for the *S* rule. Like the *VP* rule, this rule must incorporate an *NP* argument into the appropriate role in the event representation now residing in the *VP.sem*. It should, therefore, consist of another λ -application where the value of *VP.sem* provides the λ -expression and the sentence-initial *NP.sem* provides the final argument to be incorporated.

$$S \rightarrow NP \text{ VP} \quad \{VP.sem(NP.sem)\}$$

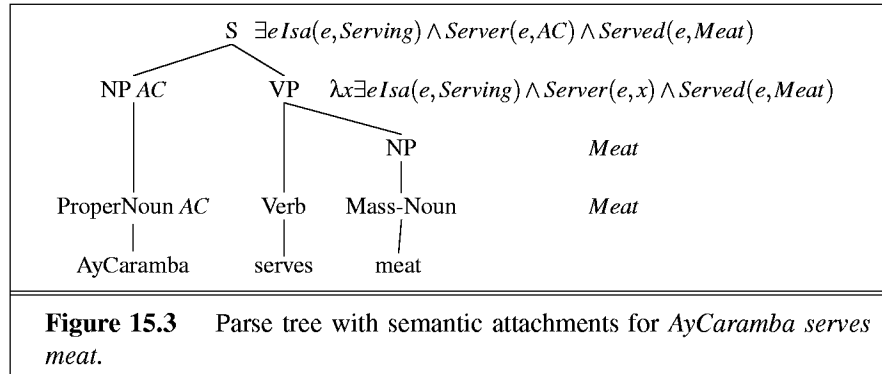
Unfortunately, as it now stands the value of *VP.Sem* doesn't provide the necessary λ expression. The *lambda*-application performed at the *VP* rule resulted in a generic FOPC expression with two existentially quantified variables. The *Verb* attachment should instead have consisted of an embedded λ -expression to make the *Server* role available for binding at the *S* level of the grammar. Therefore, our revised representation of the *Verb* attachment will be the following.

$$\begin{aligned} \text{Verb} &\rightarrow \text{serves} \\ &\quad \{\lambda x \lambda y \exists e \text{ Isa}(e, \text{Serving}) \wedge \text{Server}(e, y) \wedge \text{Served}(e, x)\} \end{aligned}$$

The body of this *Verb* attachment consists of a λ -expression inside a λ -expression. The outer expression provides the variable that is replaced by the first λ -reduction, while the inner expression can be used to bind the final variable corresponding to the *Server* role. This ordering of the variables in the multiple layers λ -expressions in semantic attachment of the verb explicitly encodes facts about the expected location of a *Verb*'s arguments in the syntax.

The parse tree for this example, with each node annotated with its corresponding semantic value, is shown in Figure 15.3.

This example has served to illustrate several of the most basic techniques used in this syntax-driven approach to semantic analysis. Section 15.2 will provide a more complete inventory of semantic attachments for some of the major English grammatical categories. Before proceeding to that inventory, however, we will first analyze several additional examples. These examples will serve to introduce a few more of the basic constructs needed to make this approach work, and will illustrate the general approach to developing semantic attachments for a grammar.



Let's consider the following variation on Example 15.1.

(15.2) A restaurant serves meat.

Since the verb phrase of this example is unchanged from Example 15.1, we can restrict our attention to the derivation of the semantics of the subject noun phrase and its subsequent integration with the verb phrase in the *S* rule. As a starting point, let's assume that the following formula is a plausible representation for the meaning of the subject in this example.

$$\exists x Isa(x, Restaurant)$$

Combining this new representation with the one already developed for the verb phrase, yields the following meaning representation.

$$\begin{aligned} &\exists e, x Isa(e, Serving) \\ &\quad \wedge Server(e, x) \wedge Served(e, Meat) \wedge Isa(x, Restaurant) \end{aligned}$$

In this formula, the restaurant, represented by the variable x , is specified as playing the role of the *Server* by its presence as the second argument to the *Server* predicate.

Unfortunately, the λ -application specified as the semantic attachment for the *S* rule will not produce this result. A literal interpretation of λ -reduction as a textual replacement results in the following expression, where the entire meaning representation of the noun phrase is embedded in the *Server* predicate.

$$\begin{aligned} &\exists e Isa(e, Serving) \\ &\quad \wedge Server(e, \exists x Isa(x, Restaurant)) \wedge Served(e, Meat) \end{aligned}$$

Although this expression has a certain intuitive appeal, it is not a valid FOPC formula. Expressions like the one denoting our restaurant can not appear as arguments to predicates; such arguments are limited to FOPC terms.

In fact, since by definition λ -expressions can only be applied to FOPC terms, the application of the λ -expression attached to the *VP* to the semantics of the subject was ill-formed to begin with.

We can solve this problem in a manner similar to the way that λ -expressions were used to solve the verb phrase and *S* semantic attachment problems: by adding a new notation to the existing FOPC syntax that facilitates the compositional creation of the desired meaning representation. In this case, we will introduce the notion of a **complex-term** that allows FOPC expressions like $\exists x \text{Isa}(x, \text{Restaurant})$ to appear in places where normally only ordinary FOPC terms would appear. Formally, a complex-term is an expression with the following three-part structure.

COMPLEX-TERM

< Quantifier variable body >

Applying this notation to our current example, we arrive at the following representation.

$$\begin{aligned} &\exists e \text{Isa}(e, \text{Serving}) \\ &\quad \wedge \text{Server}(e, < \exists x \text{Isa}(x, \text{Restaurant}) >) \wedge \text{Served}(e, \text{Meat}) \end{aligned}$$

As was the case with λ -expressions, this notational change will only be useful if we can provide a straightforward way to convert it into ordinary FOPC syntax. This can be accomplished by rewriting any predicate using a complex-term according to the following schema.

$$\begin{aligned} &P(< \text{Quantifier variable body} >) \\ &\Rightarrow \\ &\text{Quantifier variable body Connective } P(\text{variable}) \end{aligned}$$

In other words, the complex-term:

1. Is extracted from the predicate in which it appears,
2. Is replaced by the variable that represents the object in question,
3. And has its variable, quantifier, body prepended to the new expression through the use of an appropriate connective.

The following pair of expressions illustrates this complex-term reduction on our current example.

$$\begin{aligned} &\text{Server}(e, < \exists x \text{Isa}(x, \text{Restaurant}) >) \\ &\Rightarrow \\ &\exists x \text{Isa}(x, \text{Restaurant}) \wedge \text{Server}(e, x) \end{aligned}$$

The connective that is used to attach the extracted formula to the front of the new expression depends on the type of the quantifier being used: \wedge is used with \exists , and \Rightarrow is used with \forall .

It will also be useful to be able to access the three components of complex-terms. We will, therefore, extend the syntax used to refer to the semantics of a constituent by allowing reference to its parts. For example, if $A.sem$ is a complex-term then $A.sem.quantifier$, $A.sem.variable$, and $A.sem.body$ retrieve the complex-term's quantifier, variable, and body, respectively.

Returning to Example 15.2, we can now address the creation of the target meaning representation for the phrase *a restaurant*. Given the simple syntactic structure of this noun phrase, the job of the *NP* semantic attachment is fairly straightforward.

$$NP \rightarrow Det\ Nominal \quad \{ \langle Det.sem\ x\ Nominal.sem(x) \rangle \}$$

This attachment creates a complex-term consisting of a quantifier retrieved from the *Det*, followed by an arbitrary variable, and then an application of the λ -expression associated with the *Nominal* to that variable. This λ -application ensures that the correct variable appears within the predicate specified by the *Nominal*.

The attachment for the determiner simply specifies the quantifier to be used.

$$Det \rightarrow a \quad \{ \exists \}$$

The job of the nominal category is to create the *Isa* formula and λ -expression needed for use in the noun phrase.

$$Nominal \rightarrow Noun \quad \{ \lambda x Isa(x, Noun.sem) \}$$

Finally, the noun attachment simply provides the name of the category being discussed.

$$Noun \rightarrow restaurant \quad \{ Restaurant \}$$

In walking through this example, we have introduced five concrete mechanisms that instantiate the abstract functional characterization of semantic attachments that began this section.

- The association of normal FOPC expressions with lexical items.
- The association of function-like λ -expressions with lexical items.
- The copying of semantic values from children to parents.
- The function-like application of λ -expressions to the semantics of one or more children of a constituent.
- The use of complex-terms to allow quantified expressions to be temporarily treated as terms.

The introduction of λ -expressions and complex-terms was motivated by the gap between the syntax of FOPC and the syntax of English. These extra-logical devices serve to bring the syntax of FOPC closer to the syntax of the language being processed thus facilitating the semantic analysis process. Meaning representations that make use of these kinds of devices are usually referred to as **quasi-logical forms** or **intermediate representations**. Note, there is a subtle difference in usage between these two uses. The term quasi-logical form is usually applied to representations that can easily be converted to a logical representation via some simple syntactic transformation. The term intermediate representation is normally used to refer to meaning representations that serve as input to further analysis processes in an attempt to produce deeper meaning representations.

QUASI-
LOGICAL
FORMS

INTERMEDI-
ATE
REPRESENTA-
TIONS

For the purposes of this chapter, our meaning representations are quasi-logical forms since they can easily be converted to FOPC. From a somewhat broader perspective, they are also intermediate forms since further interpretation is certainly needed to get them closer to reasonable meaning representations.

The few rules introduced in this section also serve to illustrate a principle that guides the design of semantic attachments in the compositional framework. In general, it is the lexical rules that provide content level predicates and terms for our meaning representations. The semantic attachments to grammar rules put these predicates and terms together in the right ways, but do not in general introduce predicates and terms into the representation being created.

Quantifier Scoping and the Translation of Complex Terms

The schema given above to translate expressions containing complex terms into FOPC expressions is, unfortunately, not unique. Consider the following example, along with its original unscoped meaning representation.

(15.3) Every restaurant has a menu.

$$\begin{aligned} &\exists e Isa(e, Having) \\ &\quad \wedge Haver(e, < \forall x Isa(x, Restaurant) >) \\ &\quad \wedge Had(e, < \exists y Isa(y, Menu) >) \end{aligned}$$

If the complex-terms filling the *Haver* and the *Had* roles are rewritten so that the quantifier for the *Haver* role has the outer scope, then the result is the following meaning representation, which corresponds to the common-

sense interpretation of this sentence.

$$\begin{aligned} \forall x \text{Restaurant}(x) \Rightarrow \\ \exists e, y \text{Having}(e) \wedge \text{Haver}(e, x) \wedge \text{Isa}(y, \text{Menu}) \wedge \text{Had}(e, y) \end{aligned}$$

On the other hand, if the terms are rewritten in the reverse order, then the following FOPC representation results, which states that there is one menu that all restaurants share.

$$\begin{aligned} \exists y \text{Isa}(y, \text{Menu}) \wedge \forall x \text{Isa}(x, \text{Restaurant}) \Rightarrow \\ \exists e \text{Having}(e) \wedge \text{Haver}(e, x) \wedge \text{Had}(e, y) \end{aligned}$$

QUANTIFIER
SCOPING

This example illustrates the problem of ambiguous **quantifier scoping** – a single logical formula with two complex terms gives rise to two distinct and incompatible FOPC representations. In the worst case, sentences with N quantifiers will have $O(N!)$ different possible quantifier scopings.

In practice, most systems employ an ad hoc set of heuristic preference rules that can be used to generate preferred forms in order of their overall likelihood. In cases where no preference rules apply, a left to right quantifier ordering that mirrors the surface order of the quantifiers is used. Domain specific knowledge can then be used to either accept a quantified formula, or reject it and request another formula. Alshawhi (1992) presents a comprehensive approach to generating plausible quantifier scopings.

15.2 ATTACHMENTS FOR A FRAGMENT OF ENGLISH

This section describes a set of semantic attachments for a small fragment of English. As in the rest of this chapter, to keep the presentation simple, we omit the feature structures associated with these rules when they are not needed. Remember that these features are needed to ensure that the correct rules are applied in the correct situations. Most importantly for this discussion, they are needed to ensure that the correct verb entries are being employed based on their subcategorization feature structures.

Sentences

For the most part, our semantic discussions have only dealt with declarative sentences. This section expands our coverage to include the other sentence types first introduced in Chapter 9: imperatives, Yes/No questions, and WH questions. Let's start by considering the following examples.

(15.4) Flight 487 serves lunch.

(15.5) Serve lunch.

(15.6) Does Flight 207 serve lunch?

(15.7) Which flights serve lunch?

The meaning representations of these examples all contain propositions concerning the serving of lunch on flights. However, they differ with respect to the role that these propositions are intended to serve in the settings in which they are uttered. More specifically, the first example is intended to convey factual information to a hearer, the second is a request for an action, and the last two are requests for information. To capture these differences, we will introduce a set of operators that can be applied to FOPC sentences in the same way that belief operators were used in Chapter 14. Specifically, the operators *DCL*, *IMP*, *YNQ*, and *WHQ* will be applied to the FOPC representations of declaratives, imperatives, yes-no questions, and wh-questions, respectively.

Producing meaning representations that make appropriate use of these operators requires the right set of semantic attachments for each of the possible sentence types. For declarative sentences, we can simply alter the basic sentence rule we have been using as follows.

$$S \rightarrow NP VP \quad \{DCL(VP.sem(NP.sem))\}$$

The normal interpretation for a representation headed by the *DCL* operator would be as a factual statement to be added to the current knowledge-base.

Imperative sentences begin with a verb phrase and lack an overt subject. Because of the missing subject, the meaning representation for the main verb phrase will consist of a λ -expression with an unbound λ -variable representing this missing subject. To deal with this, we can simply *supply* a subject to the λ -expression by applying a final λ -reduction to a dummy constant. The *IMP* operator can then be applied to this representation as in the following semantic attachment.

$$S \rightarrow VP \quad \{IMP(VP.sem(DummyYou))\}$$

Applying this rule to Example 15.5, results in the following representation.

$$IMP(\exists e Serving(e) \wedge Server(e, DummyYou) \wedge Served(e, Lunch))$$

As will be discussed in Chapter 19, imperatives can be viewed as a kind of **speech act** – actions that are performed by virtue of being uttered.

As discussed in Chapter 9, **yes-no-questions** consist of a sentence-initial auxiliary verb, followed by a subject noun phrase and then a verb

phrase. The following semantic attachment simply ignores the auxiliary, and with the exception of the *YNQ* operator, constructs the same representation that would be created for the corresponding declarative sentence.

$$S \rightarrow Aux NP VP \quad \{YNQ(VP.sem(NP.sem))\}$$

The use of this rule with for Example 15.6 produces the following representation.

$$YNQ(\exists e Serving(e) \wedge Server(e, Flt207) \wedge Served(e, Lunch))$$

Yes-no-questions should be thought as asking the whether the propositional part of its meaning is true or false given the knowledge currently contained in the knowledge-base. Adopting the kind of semantics described in Chapter 14, yes-no-questions can be answered by determining if the proposition is in the knowledge-base, or if can be inferred from the knowledge-base.

Unlike yes-no-questions, **wh-subject-questions** ask for specific information about the subject of the sentence rather than the sentence as a whole. The following attachment produces a representation that consists of the operator *WHQ*, the variable corresponding to the subject of the sentence, and the body of the proposition.

$$S \rightarrow WhWord NP VP \quad \{WHQ(NP.sem.var, VP.sem(NP.sem))\}$$

The following representation is the result of applying this rule to Example 15.7.

$$WHQ(x, \exists e, x Isa(e, Serving) \wedge Server(e, x) \\ \wedge Served(e, Lunch) \wedge Isa(x, Flight))$$

Such questions can be answered by returning a set of assignments for the subject variable that make the resulting proposition true with respect to the current knowledge-base.

Finally, consider the following **wh-non-subject-question**.

(15.8) How can I go from Minneapolis to Long Beach?

In examples like this, the question is not about the subject of the sentence but rather some other argument, or some aspect of the proposition as a whole. In this case, the representation needs to provide an indication as to what the question is about. The following attachment provides this information by providing the semantics of the auxiliary as an argument to the *WHQ* operator.

$$S \rightarrow WhWord Aux NP VP \quad \{WHQ WhWord.sem VP.sem(NP.sem)\}$$

The following representation would result from an application of this rule to Example 15.8.

$$\begin{aligned} WHQ(How, \exists e \text{ } Isa(e, Going) \wedge Goer(e, User) \\ \wedge Origin(e, Minn) \wedge Destination(e, LongBeach)) \end{aligned}$$

As we will discuss in Section 15.5 and Chapter 19, correctly answering this kind of question involves a fair amount of domain specific reasoning. For example, the correct way to answer Example 15.8 is to search for flights with the specified departure and arrival cities. Note, however, that there is no mention of flights or flying in the actual question. The question-answerer therefore has to apply knowledge specific to this domain to the effect that questions about going places are really questions about flights to those places.

Finally, we should make it clear that this particular attachment is only useful for rather simple wh-questions without missing arguments or embedded clauses. As discussed in Chapter 11, the presence of long-distance dependencies in these questions requires additional mechanisms to determine exactly what is being asked about. Woods (1977) and Alshawi (1992) provide extensive discussions of general mechanisms for handling wh-non-subject questions. Section 15.5 presents a more ad hoc approach that is often used in practical systems.

Noun Phrases

As we have already seen, the meaning representations for noun phrases can be either normal FOPC terms or complex-terms. The following sections detail the semantic attachments needed to produce meaning representations for some of the most frequent kinds of English noun phrases. Unfortunately, as we will see, the syntax of English noun phrases provides surprisingly little insight into their meaning. It is often the case that the best we can do is provide a rather vague intermediate level of meaning representation that can serve as input to further interpretation processes.

Compound Nominals

Compound nominals, also known as noun-noun sequences, consist of simple sequences of nouns, as in the following examples.

(15.9) Flight schedule

(15.10) Summer flight schedule

As noted in Chapter 9, the syntactic structure of this construction can be captured by the regular expression *Noun**, or by the following context-free

grammar rules.

$$\textit{Nominal} \rightarrow \textit{Noun}$$

$$\textit{Nominal} \rightarrow \textit{Noun Nominal}$$

In these constructions, the final noun in the sequence is the head of the phrase and denotes an object that is semantically related in some unspecified way to the other nouns that precede it in the sequence. In general, an extremely wide range of common-sense relations can be denoted by this construction. Discerning the exact nature of these relationships is well beyond the scope of the kind of superficial semantic analysis presented in this chapter. The attachment in the following rule builds up a vague representation that simply notes the existence of a semantic relation between the head noun and the modifying nouns, by incrementally noting such a relation between the head noun and each noun to its left.

$$\textit{Nominal} \rightarrow \textit{Noun Nominal}$$

$$\{\lambda x \textit{Nominal.sem}(x) \wedge \textit{NN}(\textit{Noun.sem}, x)\}$$

The relation *NN* is used to specify that a relation holds between the modifying elements of a compound nominal and the head *Noun*. In the examples given above, this leads to the following meaning representations.

$$\lambda x \textit{Isa}(x, \textit{Schedule}) \wedge \textit{NN}(x, \textit{Flight})$$

$$\lambda x \textit{Isa}(x, \textit{Schedule}) \wedge \textit{NN}(x, \textit{Flight}) \wedge \textit{NN}(x, \textit{Summer})$$

Note that this representation correctly instantiates a term representing a *Schedule*, while avoiding the creation of terms representing either a *Flight* or *Summer*.

Genitive Noun Phrases

Recall from Chapter 9 that genitive noun phrases make use of complex determiners that consist of noun phrases with possessive markers, as in *Atlanta's airport* and *Maharani's menu*. It is quite tempting to represent the relation between these words as an abstract kind of possession. A little introspection, however, reveals that the relation between a city and its airport has little in common with a restaurant and its menu. Therefore, as with compound nominals, it turns out to be best to simply state an abstract semantic relation between the various constituents.

$$\textit{NP} \rightarrow \textit{ComplexDet Nominal}$$

$$\{< \exists x \textit{Nominal.sem}(x) \wedge \textit{GN}(x, \textit{ComplexDet.sem}) >\}$$

$$\text{ComplexDet} \rightarrow NP \text{ 's} \quad \{NP.sem\}$$

Applying these rules to *Atlanta's airport* results in the following complex term.

$$\langle \exists x Isa(x, Airport) \wedge GN(x, Atlanta) \rangle$$

Subsequent semantic interpretation would have to determine that the relation denoted by the relation *GN* is actually a location.

Adjective Phrases

English adjectives can be split into two major categories: pre-nominal and predicate. These categories are exemplified by the following BERP examples.

(15.11) I don't mind a cheap restaurant.

(15.12) This restaurant is cheap.

For the pre-nominal case, an obvious *and often incorrect* proposal for the semantic attachment is illustrated in the following rules.

$$\begin{aligned} \text{Nominal} &\rightarrow \text{Adj Nominal} \\ &\{\lambda x \text{ Nominal.sem}(x) \wedge Isa(x, \text{Adj.sem})\} \end{aligned}$$

$$\text{Adj} \rightarrow \text{cheap} \quad \{\text{Cheap}\}$$

This solution modifies the semantics of the nominal by applying the predicate provided by the adjective to the variable representing the nominal. For our cheap restaurant example, this yields the following fairly reasonable representation.

$$\lambda x Isa(x, Restaurant) \wedge Isa(x, Cheap)$$

This is an example of what is known as **intersective semantics** since the meaning of the phrase can be thought of as the intersection of the category stipulated by the nominal and the category stipulated by the adjective. In this case, this amounts to the intersection of the category of cheap things with the category of restaurants.

Unfortunately, this solution often does the wrong thing. For example, consider the following meaning representations for the phrases *small elephant*, *former friend*, and *fake gun*.

$$\lambda x Isa(x, Elephant) \wedge Isa(x, Small)$$

$$\lambda x Isa(x, Friend) \wedge Isa(x, Former)$$

$$\lambda x Isa(x, Gun) \wedge Isa(x, Fake)$$

INTERSEC-
TIVE
SEMANTICS

Each of these representations is peculiar in some way. The first one states that this particular elephant is a member of the general category of small things, which is probably not true. The second example is strange in two ways: it asserts that the person in question is a friend, which is false, and it makes use of a fairly unreasonable category of *former things*. Similarly, the third example asserts that the object in question is a gun despite the fact that *fake* means it is not one.

As with compound nominals, there is no clever solution to these problems within the bounds of our current compositional framework. Therefore, the best approach is to simply note the status of a specific kind of modification relation and assume that some further procedure with access to additional relevant knowledge can replace this vague relation with an appropriate representation (Alshaw, 1992).

$$\begin{aligned} & \text{Nominal} \rightarrow \text{Adj Nominal} \\ & \{\lambda x \text{Nominal.sem}(x) \wedge \text{AM}(x, \text{Adj.sem})\} \end{aligned}$$

Applying this rule to *a cheap restaurant* results in the following formula.

$$\exists x \text{Isa}(x, \text{Restaurant}) \wedge \text{AM}(x, \text{Cheap})$$

Note that even this watered-down proposal produces representations that are logically incorrect for the *fake* and *former* examples. In both cases, it asserts that the objects in question are in fact members of their stated categories. In general, the solution to this problem has to be based on the specific semantics of the adjectives and nouns in question. For example, the semantics of *former* has to involve some form of temporal reasoning, while *fake* requires the ability to reason about the nature of concepts and categories.

Verb Phrases

The general schema for computing the semantics of verb phrases relies on the notion of function application. In most cases, the λ -expression attached to the verb is simply applied to the semantic attachments of the verb's arguments. There are, however, a number of situations that force us to depart somewhat from this general pattern.

Infinitive Verb Phrases

A fair number of English verbs take some form of verb phrase as one of their arguments. This complicates the normal verb phrase semantic schema since these argument verb phrases interact with the other arguments of the head verb in ways that are not completely obvious.

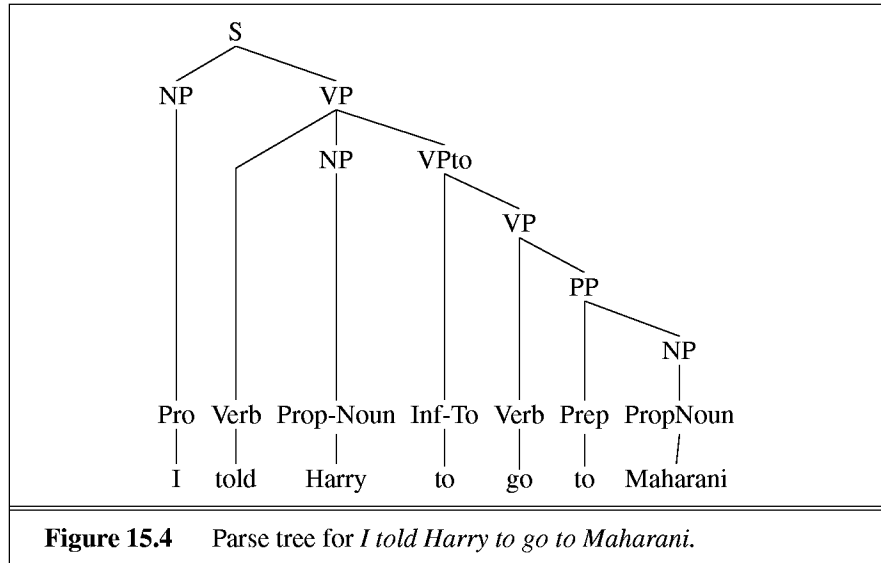


Figure 15.4 Parse tree for *I told Harry to go to Maharani.*

Consider the following example.

(15.13) I told Harry to go to Maharani.

The meaning representation for this example should be something like the the following.

$$\begin{aligned} \exists e, f, x \text{ } & Isa(e, Telling) \wedge Isa(f, Going) \\ & \wedge Teller(e, Speaker) \wedge Tellee(e, Harry) \wedge ToldThing(e, f) \\ & \wedge Goer(f, Harry) \wedge Destination(f, x) \end{aligned}$$

There are two interesting things to note about this meaning representation: the first is that it consists of two events, and the second is that one of the participants, *Harry*, plays a role in both of the two events. The difficulty in creating this complex representation falls to the verb phrase dominating the verb *tell* which will something like the following as its semantic attachment.

$$\begin{aligned} \lambda x, y \lambda z \exists e \text{ } & Isa(e, Telling) \\ & \wedge Teller(e, z) \wedge Tellee(e, x) \wedge ToldThing(e, y) \end{aligned}$$

Semantically, we can interpret this subcategorization frame for *Tell* as providing three semantic roles: a person doing the telling, a recipient of the telling, and the proposition being conveyed.

The difficult part of this example involves getting the meaning representation for the main verb phrase correct. As shown in Figure 15.2, *Harry* plays the role of both the *Tellee* of the *Telling* event and the *Goer* of the

Going event. However, *Harry* is not available when the *Going* event is created within the infinitive verb phrase.

Although there are several possible solutions to this problem, it is usually best to stick with a uniform approach to these problems. Therefore, we will start by simply applying the semantics of the verb to the semantics of the other arguments of the verb as follows.

$$VP \rightarrow Verb NP VPto \quad \{Verb.sem(NP.sem, VPto.sem)\}$$

Since the *to* in the infinitive verb phrase construction does not contribute to its meaning, we simply copy the meaning of the child verb phrase up to the infinitive verb phrase. Recall, that we are relying on the unseen feature structures to ensure that only the correct verb phrases can with this construction.

$$VPto \rightarrow to VP \quad \{VP.sem\}$$

In this solution, the verb's semantic attachment has two tasks: incorporating the *NP.sem*, the *Goer*, into the *VPto.sem*, and incorporating the *Going* event as the *ToldThing* of the *Telling*. The following attachment performs both tasks.

$$\begin{aligned} Verb &\rightarrow tell \\ &\quad \{\lambda x, y \\ &\quad \quad \lambda z \\ &\quad \quad \exists e, y.variable \text{ Isa}(e, Telling) \\ &\quad \quad \quad \wedge Teller(e, z) \wedge Tellee(e, x) \\ &\quad \quad \quad \wedge ToldThing(e, y.variable) \wedge y(x) \end{aligned}$$

In this approach, the λ -variable x plays the role of the *Tellee* of the telling and the argument to the semantics of the infinitive, which is now contained as a λ -expression in the variable y . The expression $y(x)$ represents a λ -reduction that inserts *Harry* into the *Going* event as the *Goer*. The notation $y.variable$, is analogous to the notation used for complex-term variables, and gives us access to the event variable representing the *Going* event within the infinitive's meaning representation.

Note that this approach plays fast and loose with the definition of λ -reduction, in that it allows λ -expressions to be passed as arguments to other λ -expressions, when technically only FOPC terms can serve that role. This technique is a convenience similar to the use of complex terms in that it allows us to temporarily treat complex expressions as terms during the creation of meaning representations.

Prepositional Phrases

At a fairly abstract level, prepositional phrases serve two distinct functions: they assert binary relations between their heads and the constituents to which they are attached, and they signal arguments to constituents that have an argument structure. These two functions argue for two distinct types of prepositional phrases that differ based on their semantic attachments. We will consider three places in the grammar where prepositional phrases serve these roles: modifiers of noun phrases, modifiers of verb phrases, and arguments to verb phrases.

Nominal Modifier Prepositional Phrases

Modifier prepositional phrases denote a binary relation between the concept being modified, which is external to the prepositional phrase, and the head of the prepositional phrase. Consider the following example and its associated meaning representation.

(1) A restaurant on Pearl

$$\exists x \text{ Isa}(x, \text{Restaurant}) \wedge \text{On}(x, \text{Pearl})$$

The relevant grammar rules that govern this example are the following.

$$NP \rightarrow \text{Det Nominal}$$

$$\text{Nominal} \rightarrow \text{Nominal PP}$$

$$PP \rightarrow P NP$$

Proceeding in a bottom-up fashion, the semantic attachment for this kind of relational preposition should provide a two-place predicate with its arguments distributed over two λ -expressions, as in the following.

$$P \rightarrow \text{on} \quad \{\lambda y \lambda x \text{ On}(x, y)\}$$

With this kind of arrangement, the first argument to the predicate is provided by the head of prepositional phrase and the second is provided by the constituent that the prepositional phrase is ultimately attached to. The following semantic attachment provides the first part.

$$PP \rightarrow P NP \quad \{P.\text{sem}(NP.\text{sem})\}$$

This λ -application results in a new λ -expression where the remaining argument is the inner λ -variable.

This remaining argument can be incorporated using the following nominal construction.

$$\text{Nominal} \rightarrow \text{Nominal PP} \quad \{\lambda z \text{ Nominal}.\text{sem}(z) \wedge PP.\text{sem}(z)\}$$

Verb Phrase Modifier Prepositional Phrases

The general approach to modifying verb phrases is similar to that of modifying nominals. The differences lie in the details of the modification in the verb phrase rule; the attachments for the preposition and prepositional phrase rules are unchanged. Let's consider the phrase *ate dinner in a hurry* which is governed by the following verb phrase rule.

$$VP \rightarrow VP PP$$

The meaning representation of the verb phrase constituent in this construction, *ate dinner*, is a λ -expression where the λ variable represents the as yet unseen subject.

$$\lambda x \exists e \text{ Isa}(e, \text{Eating}) \wedge \text{Eater}(e, x) \wedge \text{Eaten}(e, \text{Dinner})$$

The representation of the prepositional phrase is also a λ -expression where the λ variable is the second argument in the *PP* semantics.

$$\lambda x \text{ In}(x, < \exists h \text{ Hurry}(h) >)$$

The correct representation for the modified verb phrase should contain the conjunction of these two representations with the *Eating* event variable filling the first argument slot of the *In* expression. In addition, this modified representation must remain a λ -expression with the unbound *Eater* variable as the new λ -variable. The following attachment expression fulfills all of these requirements.

$$VP \rightarrow VP PP \quad \{\lambda y \text{ VP.sem}(y) \wedge \text{PP.sem}(\text{VP.sem.variable})\}$$

There are two aspects of this attachment that require some elaboration. The first involves the application of the constituent verb phrases' λ -expression to the variable y . Binding the lower λ -expression's variable to a new variable allows us to *lift* the lower variable to the level of the newly created λ -expression. The result of this technique is a new λ -expression with a variable that, in effect, plays the same role as the original variable in the lower expression. In this case, this allows a λ -expression to be modified during the analysis process before the argument to the expression is actually available.

The second new aspect in this attachment involves the *VP.sem.variable* notation. This notation is used to access the event-variable representing the underlying meaning of the verb phrase, in this case, e . This is analogous to the notation used to provide access the various parts of complex-terms introduced earlier.

Applying this attachment to the current example yields the following representation, which is suitable for combination with a subsequent subject noun phrase.

$$\lambda y \exists e \text{ Isa}(e, \text{Eating}) \wedge \text{Eater}(e, y) \wedge \text{Eaten}(e, \text{Dinner}) \\ \wedge \text{In}(e, < \exists h \text{Hurry}(h) >)$$

Verb Argument Prepositional Phrases

The prepositional phrases in this category serve to signal the role an argument plays in some larger event structure. As such, the preposition itself does not actually modify the meaning of the noun phrase. Consider the following example of role signaling prepositional phrases.

(15.14) I need to go from Boston to Dallas.

In examples like this, the arguments to *go* are expressed as prepositional phrases. However, the meaning representations of these phrases should consist solely of the unaltered representation of their head nouns. To handle this, argument prepositional phrases are treated in the same way that non-branching grammatical rules are; the semantic attachment of the noun phrase is copied unchanged to the semantics of the larger phrase.

$$PP \rightarrow P \text{ NP} \quad \{NP.sem\}$$

The verb phrase can then assign this meaning representation to the appropriate event role. A more complete account of how these argument bearing prepositional phrases map to underlying event roles will be presented in Chapter 16.

15.3 INTEGRATING SEMANTIC ANALYSIS INTO THE EARLEY PARSER

In Section 15.1, we suggested a simple pipeline architecture for a semantic analyzer where the results of a complete syntactic parse are passed to a semantic analyzer. The motivation for this notion stems from the fact that the compositional approach requires the syntactic parse before it can proceed. It is, however, also possible to perform semantic analysis in parallel with syntactic processing. This is possible because in our compositional framework, the meaning representation for a constituent can be created as soon as all of its constituent parts are present. This section describes just such an approach to integrating semantic analysis into the Earley parser from Chapter 10.

The integration of semantic analysis into an Earley parser is straightforward and follows precisely the same lines as the integration of unification into the algorithm given in Chapter 11. Three modifications are required to the original algorithm:

- The rules of the grammar are given a new field to contain their semantic attachments.
- The states in the chart are given a new field to hold the meaning representation of the constituent.
- The ENQUEUE function is altered so that when a complete state is entered into the chart its semantics are computed and stored in the state's semantic field.

```

procedure ENQUEUE(state, chart-entry)
  if INCOMPLETE?(state) then
    if state is not already in chart-entry then
      PUSH(state, chart-entry)
    else if UNIFY-STATE(state) succeeds then
      if APPLY-SEMANTICS(state) succeeds then
        if state is not already in chart-entry then
          PUSH(state, chart-entry)
  procedure APPLY-SEMANTICS(state)
    meaning-rep ← APPLY(state.semantic-attachment, state)
    if meaning-rep does not equal failure then
      state.meaning-rep ← meaning-rep

```

Figure 15.5 The ENQUEUE function modified to handle semantics. If the state is complete and unification succeeds then ENQUEUE calls APPLY-SEMANTICS to compute and store the meaning representation of completed states.

Figure 15.5 shows the ENQUEUE and functions modified to create meaning representations. When ENQUEUE is passed a complete state that can successfully unify its unification constraints it calls APPLY-SEMANTICS to compute and store the meaning representation for this state. Note the importance of performing feature-structure unification prior to semantic analysis. This ensures that semantic analysis will be performed only on valid trees and that features needed for semantic analysis will be present.

The primary advantage of this integrated approach over the pipeline approach lies in the fact that APPLY-SEMANTICS can fail in a manner similar to the way that unification can fail. If a semantic ill-formedness is found in the meaning representation being created, the corresponding state can be blocked from entering the chart. In this way, semantic considerations can be brought to bear during syntactic processing. Chapter 16 describes in some detail the various ways that this notion of ill-formedness can be realized.

Unfortunately, this also illustrates one of the primary disadvantages of integrating semantics directly into the parser — considerable effort may be spent on the semantic analysis of *orphan* constituents that do not in the end contribute to a successful parse. The question of whether the gains made by bringing semantics to bear early in the process outweigh the costs involved in performing extraneous semantic processing can only be answered on a case by case basis.

15.4 IDIOMS AND COMPOSITIONALITY

Ce corps qui s'appelait et qui s'appelle encore le saint empire
romain n'était en aucune manière ni saint, ni romain, ni empire.

This body, which called itself and still calls itself the Holy Roman
Empire, was neither Holy, nor Roman, nor an Empire.

– Voltaire³, 1756.

As innocuous as it seems, the principle of compositionality runs into trouble fairly quickly when real language is examined. There are many cases where the meaning of a constituent is not based on the meaning of its parts, at least not in the straightforward compositional sense. Consider the following WSJ examples.

(15.15) Coupons are just the tip of the iceberg.

(15.16) The SEC's allegations are only the tip of the iceberg.

(15.17) Coronary bypass surgery, hip replacement and intensive-care units
are but the tip of the iceberg.

The phrase *the tip of the iceberg* in each of these examples clearly doesn't have much to do with tips or icebergs. Instead, it roughly means something

³ *Essai sur les mœurs et les esprits des nations*. Translation by Y. Sills, as quoted in (Sills and Merton, 1991).

like *the beginning*. The most straightforward way to handle idiomatic constructions like these is to introduce new grammar rules specifically designed to handle them. These idiomatic rules mix lexical items with grammatical constituents, and introduce semantic content that is not derived from any of its parts.

Consider the following rule as an example of this approach.

$$NP \rightarrow \text{the tip of the iceberg} \\ \{ \text{Beginning} \}$$

The lower case items on the right-hand side of this rule are intended to represent precisely words in the input. Although, the constant *Beginning* should not be taken too seriously as a meaning representation for this idiom, it does illustrate the idea that the meaning of this idiom is not based on the meaning of any of its parts. Note that an Earley-style analyzer with this rule will now produce two parses when this phrase is encountered: one representing the idiom and one representing the compositional meaning.

Not surprisingly, as with the rest of the grammar, it may take a few tries to get to these rules right. Consider the following *iceberg* examples from the WSJ corpus.

(15.18) And that's but the tip of Mrs. Ford's iceberg.

(15.19) These comments describe only the tip of a 1,000-page iceberg.

(15.20) The 10 employees represent the merest tip of the iceberg.

The rule given above is clearly not general enough to handle these cases. These examples indicate that there is a vestigial syntactic structure to this phrase that at permits some variation in the determiners used and also permits some adjectival modification of both the *iceberg* and the *tip*. A more promising rule would be something along the following lines.

$$NP \rightarrow \text{TipNP of IcebergNP} \\ \{ \text{Beginning} \}$$

Here the categories *TipNP* and *IcebergNP* can be given an internal nominal-like structure that permits some adjectival modification and some variation in the determiners, while still restricting the heads of these noun phrases to the lexical items *tip* and *iceberg*. Note that this syntactic solution ignores the thorny issue that the modifiers *mere* and *1000-page* seem to indicate that both the *tip* and *iceberg* may in fact play some compositional role in the meaning of the idiom. We will return to this topic in Chapter 16, when we take up the issue of metaphor.

To summarize, handling idioms requires at least the following changes to the general compositional framework.

- Allow the mixing of lexical items with traditional grammatical constituents.
- Allow the creation of additional idiom-specific constituents needed to handle the correct range of productivity of the idiom.
- Permit semantic attachments that introduce logical terms and predicates that are not related to any of the constituents of the rule.

This discussion is obviously only the tip of an enormous iceberg. Idioms are far more frequent and far more productive than is generally recognized and pose serious difficulties for many applications, including as we will see in Chapter 21, machine translation.

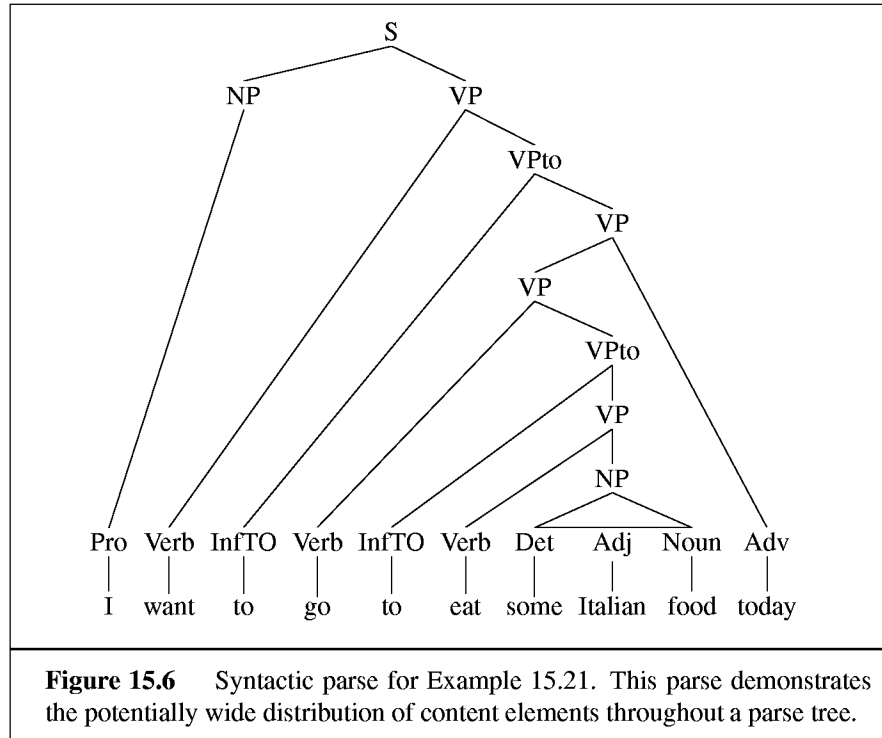
15.5 ROBUST SEMANTIC ANALYSIS

As we noted earlier, when syntax-driven semantic analysis is applied in practice, certain compromises have to be made to facilitate system development and efficiency of operation. The following sections describe the two primary ways of instantiating a syntax-driven approach in practical systems.

Semantic Grammars

When we first introduced Frege's principle of compositionality in Section 15.1, we noted that the parts referred to in that principle are the constituents provided by a syntactic grammar. Unfortunately, the syntactic structures provided by such grammars are often not particularly well-suited for the task of compositional semantic analysis. This is not particularly surprising since capturing elegant syntactic generalizations and avoiding overgeneration carry considerably more weight in the design of grammars than semantic sensibility does. This *mismatch* between the structures provided by traditional grammars and those needed for compositional semantic analysis typically manifests itself in the following three ways.

- Key semantic elements are often widely distributed across parse trees, thus complicating the composition of the required meaning representation.
- Parse trees often contain many syntactically motivated constituents that play essentially no role in semantic processing.
- The general nature of many syntactic constituents results in semantic attachments that create nearly vacuous meaning representations.



As an example of the first two problems, consider the parse tree shown in Figure 15.6 for the following BERP example.

(15.21) I want to go to eat some Italian food today.

The branching structure of this tree distributes the key components of the meaning representation widely throughout the tree. At the same time, most of the nodes in the tree contribute almost nothing to the meaning of this sentence. This structure requires three *lambda*-expressions and a complex term to bring the few contentful elements together at the top of the tree.

The third problem arises from the need to have uniform semantic attachments in the compositional rule-to-rule approach. This requirement often results in constituents that are at the right level of generality for the syntax, but too high a level for semantic purposes. A good example of this is the case of nominal compounds and adjective phrases, where the semantic attachments are so general as to be nearly meaningless. Consider, for example, the rule governing the phrase *Italian food* in our current example.

$$\begin{aligned} \text{Nominal} &\rightarrow \text{Adj Nominal} \\ &\{\lambda x \text{ Nominal.sem}(x) \wedge \text{AM}(x, \text{Adj.sem})\} \end{aligned}$$

Applying this attachment results in the following meaning representation.

$$\exists x \text{ Isa}(x, \text{Food}) \wedge \text{AM}(x, \text{Italian})$$

All nominals that fit this pattern receive the same vague interpretation that roughly indicates that the nominal is modified by the adjective. This is a far cry from what we know that expressions like *Italian food* and *Italian restaurant* mean; they denote food prepared in a particular way, and restaurants that serve food prepared that way. Unfortunately, there is no way to get this very general rule to produce such an interpretation.

Both of these problems can be overcome through the use of **semantic grammars**, which were originally developed for text-based dialog systems in the domains of question-answering and intelligent tutoring (Brown and Burton, 1975). Semantic grammars that are more directly oriented towards serving the needs of a compositional analysis. In this approach, the rules and constituents of the grammar are designed to correspond directly to entities and relations from the domain being discussed. More specifically, such grammars are constructed so that key semantic components can occur together within single rules, and rules are made no more general than is needed to achieve sensible semantic analyses.

SEMANTIC
GRAMMARS

Let's consider how these two general strategies might be applied in the BERP domain. Consider the following candidate rule for the particular kind of information request illustrated in Example 15.21.

$$\text{InfoRequest} \rightarrow \text{User want to go to eat FoodType TimeExpr}$$

As with the rules introduced for idioms, rules of this type freely mix non-terminals and terminals on their right-hand side. In this case, *User*, *FoodType*, and *TimeExpr* represent semantically motivated non-terminal categories for this domain. Given this, the semantic attachment for this rule would have all the information that it needs to compose the meaning representation for requests of this type from the immediate constituents of the rule. In particular, there is no need for λ -expressions, since this flat rule elevates all the relevant arguments to the top of the tree.

Now consider the following rule that could be used to parse the phrase *Italian food* in our example.

$$\text{FoodType} \rightarrow \text{Nationality FoodType}$$

The specific nature of this rule permits a far more useful semantic attachment than is possible with the generic nominal rule given above. More specifically, it can create a representation that states that the food specified by the con-

stituent *FoodType* is to prepared in the style associated with the *Nationality* constituent.

One of the key motivations for the use of semantic grammars in these domains was the need to deal with various kinds of anaphor and ellipsis. Semantic grammars can help with these phenomena since by their nature they enable a certain amount of *prediction*. More specifically, they allow parsers to make highly specific predictions about upcoming input, based on the categories being actively predicted by the parser. Given this ability, anaphoric references and missing elements can be associated with specific semantic categories.

As an example of how this works consider the following ATIS examples.

(15.22) When does flight 573 arrive in Atlanta?

(15.23) When does it arrive in Dallas?

Sentences like these can be analyzed with a rule like the following, which makes use of the domain specific non-terminals *Flight* and *City*.

InfoRequest → *when does Flight arrive in City*

A rule such as this gives far more information about the likely referent of the *it*, than a purely syntactic rule that would simply restrict it to anything expressible as a noun phrase. Operationally, such a system might search back in the dialog for places where the *Flight* constituent has been recently used to find candidate references for this pronoun. Chapter 18 discusses the topic of anaphor resolution in more detail.

Not surprisingly, there are a number of drawbacks to basing a system on a semantic grammar. The primary drawback arises from an almost complete lack of **reuse** in the approach. Combining the syntax and semantics of a domain into a single representation makes the resulting grammar specific to that domain. In contrast, systems that keep their syntax and semantics separate can, in principle, reuse their grammars in new domains. A second lack of reuse arises as a consequence of eschewing syntactic generalizations in the grammar. This results in an unavoidable growth in the size of the grammar for a single domain. As an example of this, consider that whereas our original noun phrase rule was sufficient to cover both *Italian restaurant* as well as *Italian food*, we now need two separate rules for these phrases. In fact, inspection of the BERP corpus reveals that we would also need also need additional rules for *vegetarian restaurant*, *California restaurant*, and *expensive restaurant*.

We should also note that semantic grammars are susceptible to a kind of semantic overgeneration. As an example of this, consider the phrase *Canadian restaurant*. It matches the rule given above for ethnic restaurants, and would result in a meaning representation that specifies a restaurant that serves food prepared in the Canadian style. Unfortunately, this is almost certainly an incorrect interpretation of this phrase; none of the occurrences of this phrase in the WSJ corpus had this meaning, all referring instead to restaurants located within Canada. Dialog systems that use semantic grammars rely on the rarity of such uses in restricted domains.

Finally, we should note that semantic grammars probably should have been called something else, since in practice the grammars themselves are formally the same as any other grammar formalism we have discussed in this book. Correspondingly, there are no special algorithms for syntactic or semantic analysis specific to semantic grammars; they can use whatever algorithms are appropriate for the grammar formalism being employed, such as Earley, or any other context-free parsing algorithm.

Information Extraction

In language processing tasks such question-answering, coming to a reasonable understanding of each input sentence is vital since giving a user a wrong answer can have serious consequences. For these tasks, the rule-to-rule approach with an eye towards semantics is a good way to build a complete interpretation of an input sentence.

However, other tasks, like extracting information about joint ventures from business news, understanding weather reports, or summarizing simple information about what happened today on the stock market from a radio report, do not necessarily require this kind of detailed understanding. Such **information extraction** tasks are characterized by two properties: (1) the desired knowledge can be described by a relatively simple and fixed **template**, or frame, with slots that need to be filled in with material from the text, and (2) only a small part of the information in the text is relevant for filling in this frame; the rest can be ignored.

For example, one of the tasks used in the fifth *Message Understanding Conference* (MUC-5) in 1993 (Sundheim, 1993), a U.S. Government-organized information extraction conference, was to extract information about international joint ventures from business news. Here are the first two sentences of a sample article from (Grishman and Sundheim, 1995):

Bridgestone Sports Co. said Friday it has set up a joint venture in Tai-

INFORMATION
EXTRACTION

TEMPLATE

METHODOLOGY BOX: EVALUATING INFORMATION EXTRACTION SYSTEMS

The information extraction paradigm has much in common with the field of information retrieval and has adapted several standard evaluation metrics from information retrieval including **precision**, **recall**, **fallout**, and a combined metric called an **F-measure**.

Recall is a measure of how much relevant information the system has extracted from the text; it is thus a measure of the coverage of the system. Recall is defined as follows:

$$\text{Recall} = \frac{\text{\# of correct answers given by system}}{\text{total \# of possible correct answers in the text}}$$

Precision is a measure of how much of the information that the system returned is actually correct, and is also known as **accuracy**. Precision is defined as follows:

$$\text{Precision} = \frac{\text{\# of correct answers given by system}}{\text{\# of answers given by system}}$$

Fallout is a measure of the systems ability to ignore spurious information in the text. It is defined as follows:

$$\text{Fallout} = \frac{\text{\# of incorrect answers given by system}}{\text{\# of spurious facts in the text}}$$

Note that recall and precision are antagonistic to one another since a conservative system that strives for perfection in terms of precision will invariably lower its recall score. Similarly, a system that strives for coverage will get more things wrong, thus lowering its precision score. This situation has led to the use of a combined measure called the **F-measure** that balances recall and precision by using a parameter β . The F-measure is defined as follows:

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

When β is one, precision and recall are given equal weight. When β is greater than one, precision is favored, and when β is less than one, recall is favored.

TIE-UP-1:	
Relationship:	TIE-UP
Entities:	"Bridgestone Sports Co." "a local concern" "a Japanese trading house"
Joint Venture Company	"Bridgestone Sports Taiwan Co."
Activity	ACTIVITY-1
Amount	NT\$20000000
ACTIVITY-1:	
Company	"Bridgestone Sports Taiwan Co."
Product	"iron and "metal wood" clubs"
Start Date	DURING: January 1990

Figure 15.7 The templates produced by the FASTUS (Hobbs *et al.*, 1997) information extraction engine given the input text on page 575.

wan with a local concern and a Japanese trading house to produce golf clubs to be shipped to Japan.

The joint venture, Bridgestone Sports Taiwan Co., capitalized at 20 million new Taiwan dollars, will start production in January 1990 with production of 20,000 iron and "metal wood" clubs a month.

The output of an information extraction system can be a single template with a certain number of slots filled in, or a more complex hierarchically related set of objects. The MUC-5 task specified this latter more complex output, requiring systems to produce hierarchically linked templates describing the participants in the joint venture, the resulting company, and its intended activity, ownership and capitalization. Figure 15.7 shows the resulting structure produced by the FASTUS system (Hobbs *et al.*, 1997).

Many information extraction systems are built around **cascades** of finite-state automata. The FASTUS system, for example, produces the template given above, based on a cascade in which each level of linguistic processing extracts some information from the text, which is passed on to the next higher level, as shown in Figure 15.8

Many systems base all or most of these levels on finite-automata, although in practice most complete systems are not technically finite-state, either because the individual automata are augmented with feature registers (as in FASTUS), or because they are used only as preprocessing steps for full parsers (e.g. Gaizauskas *et al.*, 1995; Weischedel, 1995) indexGaizauskas, R.), or are combined with other components based on decision-trees (Fisher

No.	Step	Description
1	Tokens:	Transfer an input stream of characters into a token sequence.
2	Complex Words:	Recognize multi-word phrases, numbers, and proper names.
3	Basic phrases:	Segment sentences into noun groups, verb groups, and particles.
4	Complex phrases:	Identify complex noun groups and complex verb groups.
5	Semantic Patterns:	Identify semantic entities and events and insert into templates.
6	Merging:	Merge references to the same entity or event from different parts of the text.
Figure 15.8 Levels of processing in FASTUS(Hobbs <i>et al.</i> , 1997). Each level extracts a specific type of information which is then passed on to the next higher level.		

et al., 1995).

Let's sketch the FASTUS implementation of each of these levels, following Hobbs *et al.* (1997) and Appelt *et al.* (1995). After tokenization, the second level recognizes multiwords like *set up*, and *joint venture*, and names like *Bridgestone Sports Co.*. The name recognizer is a transducer, composed of a large set of specific mappings designed to handle locations, personal names, and names of organizations, companies, unions, performing groups, etc. The following are typical rules for modeling names of performing organizations like *San Francisco Symphony Orchestra* and *Canadian Opera Company*. While the rules are written using a context-free syntax, there is no recursion and therefore they can be automatically compiled into finite-state transducers:

Performer-Org	→ (pre-location) Performer-Noun+ Perf-Org-Suffix
pre-location	→ locname nationality
locname	→ city region
Perf-Org-Suffix	→ orchestra, company
Performer-Noun	→ symphony, opera
nationality	→ Canadian, American, Mexican
city	→ San Francisco, London

The second stage also might transduce sequences like *forty two* into

the appropriate numeric value (recall the discussion of this problem on page 124 in Chapter 5).

The third FASTUS stage produces a series of **basic phrases**, such as noun groups, verb groups, etc., using finite-state rules of the sort shown on page 386. The output of the FASTUS basic phrase identifier is shown in Figure 15.9; note the use of some domain-specific basic phrases like *Company* and *Location*.

BASIC
PHRASES

Company	Bridgestone Sports Co.
Verb Group	said
Noun Group	Friday
Noun Group	it
Verb Group	had set up
Noun Group	a joint venture
Preposition	in
Location	Taiwan
Preposition	with
Noun Group	a local concern
Conjunction	and
Noun Group	a Japanese trading house
Verb Group	to produce
Noun Group	golf clubs
Verb Group	to be shipped
Preposition	to
Location	Japan

Figure 15.9 The output of Stage 2 of the FASTUS basic-phrase extractor, which uses finite-state rules of the sort described by Appelt and Israel (1997) and shown on page 386.

Recall that Chapter 10 described how these basic phrases can be combined into complex noun groups and verb groups. This is accomplished in Stage 4 of FASTUS, by dealing with conjunction and with the attachment of measure phrases as in the following.

20,000 iron and "metal wood" clubs a month,
and preposition phrases:

production of 20,000 iron and "metal wood" clubs a month,

The output of Stage 4 is a list of complex noun groups and verb groups. Stage 5 takes this list, ignoring all input that has not been chunked into a complex group, recognizes entities and events in the complex groups, and inserts the recognized objects into the proper templates. The recognition of

(1)	Relationship: Entities:	TIE-UP “Bridgestone Sports Co.” “a local concern” “a Japanese trading house”
(2)	Activity Product	PRODUCTION “golf clubs”
(3)	Relationship: Joint Venture Company Amount	TIE-UP “Bridgestone Sports Taiwan Co.” NT\$20000000
(4)	Activity Company Start Date	PRODUCTION “Bridgestone Sports Taiwan Co.” DURING: January 1990
(5)	Activity Product	PRODUCTION “iron and “metal wood” clubs”
Figure 15.10 The five partial templates produced by Stage 5 of the FASTUS system. These templates will be merged by the Stage 6 Merging algorithm to produce the final template shown in Figure 15.7 on page 577.		

entities and events is done by hand-coded finite-state automata whose transitions are based on particular complex-phrase types annotated by particular head words or particular features like *company*, *currency*, or *date*.

For example, the first sentence of the news story above realizes the semantic patterns based on the following two regular expressions (where NG indicates Noun-Group and VG Verb-Group).

- NG(Company/ies) VG(Set-up) NG(Joint-Venture) with NG(Company/ies)
- VG(Produce) NG(Product)

The second sentence realizes the second pattern above as well as the following two patterns:

- NG(Company) VG-Passive(Capitalized) at NG(Currency)
- NG(Company) VG(Start) NG(Activity) in/on NG(Date)

The result of processing these two sentences is the set of five draft templates shown in Figure 15.10. These five templates must then be merged into the single hierarchical structure shown in Figure 15.7. The merging algorithm decides whether two activity or relationship structures are sufficiently consistent that they might be describing the same events, and merges them if so. Since the merging algorithm must perform reference resolution (deciding when it is the case that two descriptions refer to the same entity), we defer description of this level to Chapter 18.

Domain-specific templates of the kind we have described in this section have also been used in many limited-domain semantic understanding and discourse comprehension tasks, including managing mixed dialog in question-answering systems (Bobrow *et al.*, 1977).

15.6 SUMMARY

This chapter explores the notion of syntax-driven semantic analysis. Among the highlights of this chapter are the following topics.

- Semantic analysis is the process whereby meaning representations are created and assigned to linguistic inputs.
- Semantic analyzers that make use of static knowledge from the lexicon and grammar can create context independent literal, or conventional, meanings.
- The *Principle of Compositionality* states that the meaning of a sentence can be composed from the meanings of its parts.
- In syntax-driven semantic analysis, the parts are the syntactic constituents on an input.
- Compositional creation of FOPC formulas is possible with a few notational extensions including λ -expressions and complex terms.
- Natural language quantifiers introduce a kind of ambiguity that is difficult to handle compositionally. Complex terms can be used to compactly encode this ambiguity.
- Idiomatic language defies the principle of compositionality but can easily be handled by adapting the techniques used to design grammar rules and their semantic attachments.
- Practical semantic analysis systems adapt the strictly compositional approach in a number of ways.
 - Dialog systems based on semantic grammars rely on grammars that have been written to serve the needs of semantics rather than syntactic generality.
 - Information extraction systems based on cascaded automata can extract pertinent information while ignoring irrelevant parts of the input.

BIBLIOGRAPHICAL AND HISTORICAL NOTES

As noted earlier, the principle of compositionality is traditionally attributed to Frege; Janssen (1997) discusses this attribution. Using the categorial grammar framework described in Chapter 12, Montague (1973) demonstrated that a compositional approach could be systematically applied to an interesting fragment of natural language. The rule-to-rule hypothesis was first articulated by (Bach, 1976). On the computational side of things, Woods's LUNAR system (Woods, 1977) was based on a pipelined syntax-first compositional analysis. Schubert and Pelletier (1982) developed an incremental rule-to-rule system based on Gazdar's GPSG approach (Gazdar, 1981, 1982; Gazdar *et al.*, 1985). Main and Benson (1983) extended Montague's approach to the domain of question-answering.

In one of the all too frequent cases of parallel development, researchers in programming languages developed essentially identical compositional techniques to aid in the design of compilers. Specifically, Knuth (1968) introduced the notion of attribute grammars that associate semantic structures with syntactic structures in a one to one correspondence. As a consequence, the style of semantic attachments used in this chapter will be familiar to users of the YACC-style (Johnson and Lesk, 1978) compiler tools.

Semantic Grammars are due to Burton (Brown and Burton, 1975). Similar notions developed around the same time included Pragmatic Grammars (Woods, 1977), and Performance Grammars (Robinson, 1975). All centered around the notion of reshaping syntactic grammars to serve the needs of semantic processing. It is safe to say that most modern systems developed for use in limited domains make use of some form of semantic grammar.

Most of the techniques used in the fragment of English presented in Section 15.2 are adapted from SRI's Core Language Engine (Alshaw, 1992). Additional bits and pieces were adapted from (Woods, 1977; Schubert and Pelletier, 1982; Gazdar *et al.*, 1985). Of necessity, a large number of important topics were not covered in this chapter. See (Alshaw, 1992) for the standard gap-threading approach to semantic interpretation in the presence of long-distance dependencies. ter Meulen (1995) presents an up to date treatment of tense, aspect, and the representation of temporal information. Extensive coverage of approaches to quantifier scoping can be found in (Hobbs and Shieber, 1987; Alshaw, 1992). van Lehn (1978) presents a

set of human preferences for quantifier scoping. Over the years, a considerable amount of effort has been directed toward the interpretation of nominal compounds. Linguistic research on this topic can be found in (Lees, 1970; Downing, 1977; Levi, 1978; Ryder, 1994), more computational approaches are described in (Gershman, 1977; Finin, 1980; McDonald, 1982; Pierre, 1984; Arens *et al.*, 1987; Wu, 1992; Vanderwende, 1994; Lauer, 1995).

There is a long and extensive literature on idioms. Fillmore *et al.* (1988) describe a general grammatical framework that places idioms at the center of its underlying theory. Makkai (1972) presents an extensive linguistic analysis of many English idioms. Hundreds of idiom dictionaries for second language learners are also available. On the computational side, Becker (1975) was among the first to suggest the use of phrasal rules in parsers. Wilensky and Arens (1980) were among the first to successfully make use of this notion. Zernik (1987) demonstrated a system that could learn such phrasal idioms in context. A collection of papers on computational approaches to idioms appeared in (Fass *et al.*, 1992).

The first work on information extraction was performed in the context of the Frump system (DeJong, 1982). Later work was stimulated by the U.S government sponsored MUC conferences (Sundheim, 1991, 1992, 1993, 1995b). Chinchor *et al.* (1993) describes the evaluation techniques used in the MUC-3 and MUC-4 conferences. Hobbs (1997) partially credits the inspiration for FASTUS to the success of the University of Massachusetts CIRCUS system (Lehnert *et al.*, 1991) in MUC-3. The SCISOR system is another system based loosely on cascades and semantic expectations that did well in MUC-3 (Jacobs and Rau, 1990). Due to the lack of reuse from one domain to another in information extraction, a considerable amount of work has focused on automating the process of knowledge acquisition in this area. A variety of supervised learning approaches are described in (Cardie, 1993, 1994; Riloff, 1993; Soderland *et al.*, 1995; Huffman, 1996; Freitag, 1998).

Finally, we have skipped an entire branch of semantic analysis in which expectations driven from deep meaning representations drive the analysis process. Such systems avoid the direct representation and use of syntax, rarely making use of anything resembling a parse tree. The earliest and most successful efforts along these lines were developed by Simmons (1973b, 1978, 1983) and (Wilks, 1975a, 1975c). A series of similar approaches were developed by Roger Schank and his students (Riesbeck, 1975; Birnbaum and Selfridge, 1981; Riesbeck, 1986). In these approaches, the semantic analysis process is guided by detailed procedures associated with individual lexical

items. The CIRCUS information extraction system (Lehnert *et al.*, 1991) traces its roots to these systems.

EXERCISES

15.1 The attachment given on page 560 to handle noun phrases with complex determiners is not general enough to handle most possessive noun phrases. Specifically, it doesn't work for phrases like the following.

- a. My sister's flight
- b. My fiance's mother's flight

Create a new set of semantic attachments to handle cases like these.

15.2 Develop a set of grammar rules and semantic attachments to handle predicate adjectives such as the one following.

- a. Flight 308 from New York is expensive.
- b. Murphy's restaurant is cheap.

15.3 None of the attachments given in this chapter provide temporal information. Augment a small number of the most basic rules to add temporal information along the lines sketched in Chapter 14. Use your rules to create meaning representations for the following examples.

- a. Flight 299 departed at 9 o'clock.
- b. Flight 208 will arrive at 3 o'clock.
- c. Flight 1405 will arrive late.

15.4 As noted in Chapter 14, the present tense in English can be used to refer to either the present or the future. However, it can also be used to express habitual behavior, as in the following.

Flight 208 leaves at 3 o'clock.

This could be a simple statement about today's Flight 208, or alternatively it might state that this flight leaves at 3 o'clock every day. Create a

FOPC meaning representation along with appropriate semantic attachments for this habitual sense.

15.5 Implement the Earley-based semantic analyzer described in Section 15.3.

15.6 It has been claimed that it is not necessary to explicitly list the semantic attachment for most grammar rules. Instead, the semantic attachment for a rule should be inferable from the semantic types of the rule's constituents. For example, if a rule has two constituents where one is a single argument λ -expression and the other is a constant then the semantic attachment should obviously apply the λ -expression to the constant. Given the attachments presented in this chapter, does this *type-driven semantics* seem like a reasonable idea?

15.7 Add a simple type-driven semantics mechanism to the Earley analyzer you implemented for Exercise 15.5

15.8 Using a phrasal search on your favorite Web search engine, collect a small corpus of *the tip of the iceberg* examples. Be certain that you search for an appropriate range of examples (ie. don't just search of "the tip of the iceberg".) Analyze these examples and come up with a set of grammar rules that correctly accounts for them.

15.9 Collect a similar corpus of examples for the idiom *miss the boat*. Analyze these examples and come up with a set of grammar rules that correctly accounts for them.

15.10 There are now a fair number of Web-based natural language question answering services that purport to provide answers to questions on a wide range of topics (see this book's Web page for pointers to current services.) Develop a corpus of questions for some general domain of interest and use it to evaluate one or more of these services. Report your results. What difficulties did you encounter in applying the standard evaluation techniques to this task?

15.11 Collect a small corpus of weather reports from your local newspaper or the Web. Based on an analysis of this corpus, create a set of frames sufficient to capture the semantic content of these reports.

15.12 Implement and evaluate a small information extraction system for the weather report corpus you collected for the last exercise.

16

LEXICAL SEMANTICS

‘When I use a word,’ Humpty Dumpty said in rather a scornful tone, ‘it means just what I choose it to mean – neither more nor less.’

Lewis Carroll’s *Alice in Wonderland*

How many legs does a dog have if you call its tail a leg?

Four.

Calling a tail a leg doesn’t make it one.

Attributed to Abraham Lincoln

A revised version of this chapter will be available shortly.

The previous two chapters focused on representing and creating meaning representations for entire sentences. In those discussions, we made minimal use of the notion of the *meaning of a word*. Words and their meanings were of interest solely to the extent that they provided the appropriate bits and pieces necessary to construct adequate meaning representations for entire sentences. This general approach is motivated by the view that while words may contribute content to the meanings of sentences, they do not themselves have meanings. By this we mean that words, by themselves, do not refer to the world, can not be judged to be true or false, or literal or figurative, or a host of other things that are generally reserved to entire sentences and utterances. This narrow conception of the role of words in a semantic theory leads to a view of the lexicon as a simple listing of symbolic fragments devoid of any systematic structure.

LEXICAL
SEMANTICS

The topics presented in this chapter serve to illustrate how much is missed by this narrow view. As we will see, the lexicon has a highly systematic structure that governs what words can mean, and how they can be used. This structure consists of relations among words and their meanings, as well as the internal structure of individual words. The study of this systematic, meaning related, structure is called **Lexical Semantics**.

LEXEME

Before moving on, we will first introduce a few new terms, since the ones we have been using thus far are entirely too vague. In particular, the word *word* has by now been used in so many different ways that it will prove difficult to make unambiguous use of it in this chapter. Instead, we will focus on the notion of a **lexeme**, an individual entry in the lexicon. A lexeme should be thought of as a pairing of a particular orthographic and phonological form with some form of symbolic meaning representation. The **lexicon** is therefore a finite list made up of lexemes. When appropriate, we will use the terms orthographic form, and phonological form, to refer to the appropriate form part of this pairing, and the term **sense** to refer to a lexeme's meaning component. Note that these definitions will undergo a number of refinements as needed in later sections.

SENSE

Given this minimal nomenclature, let us return to the topic of what facts we can discover about lexemes that are relevant to the topic of meaning. A fruitful place to start such an exploration is a dictionary. Dictionaries are, after all, nothing if not repositories of information about the meanings of lexemes. Within dictionaries, it turns out that the most interesting place to look first is at the definitions of lexemes that no one ever actually looks up. For example, consider the following fragments from the definitions of *right*, *left*, *red*, *blood* from the *American Heritage Dictionary* (Morris, 1985).

right *adj* located nearer the right hand esp. being on the right when facing the same direction as the observer.

left *adj* located nearer to this side of the body than the right.

red *n* the color of blood or a ruby.

blood *n* the red liquid that circulates in the heart, arteries and veins of animals.

The first thing to note about these definitions is the surprising amount of circularity in them. The definition of *right* makes two direct references to itself, while the entry for *left* contains an implicit self-reference in the phrase *this side of the body*, which presumably means the *left* side. The entries for *red* and *blood* avoid this kind of direct self-reference by instead referencing each other in their definitions. Such circularity is, of course, inherent in all dictionary definitions, these examples are just extreme cases. In the end, all

definitions are stated in terms of lexemes that are, in turn, defined in terms of other lexemes.

From a purely formal point of view, this inherent circularity is evidence that what dictionaries entries provide are not, in fact, definitions at all. They are simply descriptions of lexemes in terms of other lexemes, with the hope being that the user of the dictionary has sufficient grasp of these other terms to make the entry in question sensible. As is obvious with lexemes like *red* and *right*, this approach will fail without some ultimate grounding in the external world.

Fortunately, even with this limitation, there is still a wealth of semantic information contained in these kinds of definitions. For example, the above definitions make it clear that *right* and *left* are similar kinds of lexemes that stand in some kind of alternation, or opposition, to one another. Similarly, we can glean that *red* is a color, it can be applied to both *blood* and *rubies*, and that *blood* is a *liquid*. As we will see in this chapter, given a sufficiently large database of facts such as these, many applications are quite capable of performing sophisticated semantic tasks (even if they do not *really* know their right from their left.)

To summarize, we can capture quite a bit about the semantics of individual lexemes by analyzing and labeling their relations to other lexemes in various settings. We will, in particular, be interested in accounting for the similarities and differences among different lexemes in similar settings, and the nature of the relations among lexemes in a single setting. This latter topic will lead us to examine the idea that lexemes are not unanalyzable atomic symbols, but rather have an internal structure that governs their combinatoric possibilities. Later, in Section 16.4, we will take a closer look at the notion of creativity, or generativity, and the lexicon. There we will explore the notion that the lexicon should not be thought of as a finite listing, but rather as a creative generator of infinite meanings.

Before proceeding, we should note that the view of lexical semantics presented here is not oriented solely towards improving computational applications of the more restrictive “only sentences have meaning” variety. Rather, as we will see, it lends itself to a wide array of applications that involve the use of words, and that could be improved by some knowledge of their meanings.

16.1 RELATIONS AMONG LEXEMES AND THEIR SENSES

The section explores a variety of relations that hold among lexemes and among their senses. The list of relations presented here is by no means exhaustive; the emphasis is on those relations that have had significant computational implications. As we will see, the primary analytic tool we will use involves the systematic substitution of one lexeme for another in some setting. The results of such substitutions can reveal the presence or absence of a specific relationship between the substituted lexemes.

Homonymy

HOMONYMY We begin this section with a discussion of **homonymy**, perhaps the simplest, and semantically least interesting, relation to hold between lexemes. Traditionally, homonymy is defined as a relation that holds *between words that have the same form with unrelated meanings*. The items taking part in such a relation are called **homonyms**. HOMONYMS A classic example of homonymy is *bank* with its distinct financial institution and sloping mound meanings, as illustrated in the following WSJ examples.

(16.1) Instead, a *bank* can hold the investments in a custodial account in the client's name.

(16.2) But as agriculture burgeons on the east *bank*, the river will shrink even more.

Loosely following lexicographic tradition, we will denote this relationship by placing a superscript on the orthographic form of the word as in **bank**¹ and **bank**². This notation indicates that these are two separate lexemes, with distinct and unrelated meanings, that happen to share an orthographic form.

It will come as little surprise that any definition this simple will prove to be problematic and will need to be refined. In the following discussion, we will explore this definition by examining pairs of words that satisfy it, but which for a number of reasons seem to be marginal examples. We will begin by focusing solely on issues of form, returning later to the topic of meaning. Note that while this may seem like an odd choice given the topic of this chapter, these discussions will serve to introduce a number of important distinctions needed in later sections. In this discussion, we will be primarily concerned with how well our definition of homonymy assists us in identifying and characterizing those lexemes which will lead to ambiguity problems for various applications.

Returning to the *bank* example, the first thing to note is that **bank**¹ and **bank**² are identical in both their orthographic *and* phonological forms. Of course, there are also pairs of lexemes with distinct meanings which do not share *both* forms. For example, pairs like *wood* and *would*, and *be* and *bee*, are pronounced the same but are spelled differently. Indeed, as we saw in Chapter 5, when pronunciation in context is taken into account, the situation is even worse. Recall, that the lexemes *knee*, *need*, *neat*, *new*, *you*, *the*, and *to* can all be pronounced as [ni], given the right context. Clearly, if the notion of form in our definition of homonymy includes a word's phonological form in context, there will be a huge number of homonyms in English.

Of course, none of these examples are traditionally be considered good candidates for homonymy. The notion of homonymy is most closely associated with the field of lexicography, where normally only dictionary entries with identical **citation-forms** are considered candidates for homonymy. Citation-forms are the orthographic-forms that are used to alphabetically index words in a dictionary, which in English correspond to what we have been calling the root form of a word. Under this view, words with the same pronunciation but different spellings are not considered homonyms, but rather **homophones**, distinct lexemes with a shared pronunciation.

CITATION-FORMS

HOMO-PHONES

Of course, there are also pairs of lexemes with identical orthographic forms with different pronunciations. Consider, for example, the distinct fish and music meanings associated with the orthographic form *bass* in the following examples.

(16.3) The expert angler from Dora, Mo., was fly-casting for bass rather than the traditional trout.

(16.4) The curtain rises to the sound of angry dogs baying and ominous bass chords sounding.

While these examples more closely fit the traditional definition of homonymy, they would only rarely appear in any traditional list of homonyms. Instead, lexemes with the same orthographic form with unrelated meanings are called **homographs**.

HOMO-GRAPHS

Finally, we should note that lexemes with different parts of speech are also typically not considered to be good candidates for homonymy. This restriction serves to rule out examples such as *would* and *wood*, on grounds other than their orthography. The basis for this restriction is two-fold: first as we saw when we discussed part-of-speech tagging, lexemes with such different parts of speech are easily distinguished based on their differing syntactic environments, and secondly lexical items can take on many distinct

forms based on their inflectional and derivational morphology, which is in turn largely based on part-of-speech.

To complicate matters, the issue of differing morphology can also occur with lexemes that have the same part-of-speech. Consider the lexemes *find* and *found* in their *locating* and *creating an institution* meanings, as illustrated in the following WSJ examples.

- (16.5) He has looked at 14 baseball and football stadiums and found that only one - - private Dodger Stadium – brought more money into a city than it took out.
- (16.6) Culturally speaking, this city has increasingly displayed its determination to found the sort of institutions that attract the esteem of Eastern urbanites.

Here we have two lexemes with distinct root forms, *find* and *found*, that nevertheless share the morphological variant *found* as the past tense of the first, and the root of the second.

At this point, having raised all of these complexities, we might create a more refined definition for homonymy as two lexemes with unrelated meanings, the same part of speech, and identical orthographic and phonological forms in all possible morphological derivations. Under this definition, all homonyms would also be both homographs and homophones, with the converse not necessarily being the case. Under this new definition, most of the homographs and homophones presented earlier would be ruled out as homonyms.

Such definitional exercises, however, merely obscure our reason for raising the issue of homonymy in the first place; homonymy is of interest computationally to the extent that it leads an application into dealing with ambiguity. Whether or not a given pair of lexemes cause ambiguity to arise in an application is entirely dependent on the nature of the application. As we will see in the following discussion of various applications, distinguishing perfect examples of homonymy from imperfect examples is of very little practical value. The critical issue is whether the nature of the form overlap is likely to cause difficulties for a given application.

In **spelling correction**, homophones can lead to real-word spelling errors, or malapropisms, as when lexemes such as *weather* and *whether* are interchanged. Note that this is a case where a phonological overlap causes a problem for a purely text-based system. Additional problems in spelling correction are caused by such imperfect homographs as *find* and *found*, which have partially overlapping morphologies. In this case, a word-form like

founded may represent a correct use of the past tense, or an incorrect over-application of the regular past tense rule to an irregular verb.

In **speech recognition**, homophones such as *to*, *two* and *too* cause obvious problems. What is less clear, however, is that perfect homonyms such as *bank* are also problematic. Recall that speech recognition systems rely on language models that are often based on tables of N-gram probabilities. For perfect homonyms, the entries for all the distinct lexemes are conflated despite the fact that the different lexemes occur in different environments. This conflation results in inappropriately high probabilities to words that are cohorts of the lexeme not in use, and lower than appropriate probabilities to the correct cohorts.

Finally, **text-to-speech** systems are vulnerable to homographs with distinct pronunciations. This problem can be avoided to some extent with examples such as *conduct* whose different pronunciations are associated with the distinct parts of speech through the use of part-of-speech tagging. However, for other examples like *bass* the two lexemes must be distinguished by some other means. Note that this situation is the reverse of the one we had with spelling correction, here a fundamentally speech-oriented system is being plagued by an orthographic problem.

Polysemy

Having muddled the waters discussing issues of form and homonymy, let us return to the topic of what it means for two meanings to be related or unrelated. Recall that the definition of homonymy requires that the lexemes in question have distinct and unrelated meanings. This is the crux of the matter; if the meanings in question are related in some way then we are dealing with a single lexeme with more than one meaning, rather than two separate lexemes. This phenomenon of a single lexeme with multiple related meanings is known as **polysemy**. Note that earlier we had defined a lexeme as a pairing between a surface form and a sense. Here we will expand that notion to be a pairing of a form with a set of related senses.

POLYSEMY

To make this notion more concrete, consider the following *bank* example from the WSJ corpus.

(16.7) While some *banks* furnish sperm only to married women, others are much less restrictive.

Although this is clearly not a use of the sloping mound meaning of *bank*, it just as clearly is not a reference to a promotional giveaway at a financial institution. One way to deal with this use would be to create **bank³**, yet

another distinct lexeme associated with the form *bank*, and give it a meaning appropriate to this use. Unfortunately, according to our definition of homonymy, this would require us to say that the meaning of *bank* in this example is distinct and unrelated to the financial institution sense, which seems to be far too strong a statement. The notion of polysemy allows us to state that this sense of *bank* is related to, and possibly derived from, the financial institution sense, without asserting that it is a distinct lexeme.

As one might suspect, the task of distinguishing homonymy from polysemy is not quite as straightforward as we made it seem with these *bank* examples. There are two criteria that are typically invoked to determine whether or not the meanings of two lexemes are related or not: the history, or **etymology**, of the lexemes in question, and how the words are conceived of by native speakers. In practice, an ill-defined combination of evidence from these two sources is used to distinguish homonymous from polysemous lexical entries. In the case of *bank*, the etymology reveals that **bank**¹ has an Italian origin, while **bank**² is of Scandinavian origin, thus encouraging us to list them as distinct lexemes. On the other hand, our belief that the use of *bank* in Example 16.7 is related to **bank**¹ is based on introspection about the similarities of their meanings, and the lack of any etymological evidence for an independent third sense.

In the absence of detailed etymological evidence, a useful intuition to use in distinguishing homonymy from polysemy is the notion of coincidence. Cases of homonymy can usually be understood easily as accidents of history – two lexemes which have coincidentally come to share the same form. On the other hand, it is far more difficult to accept cases of polysemy as coincidences. Returning again to our *bank* example, it is difficult to accept the idea that the various uses of *bank* in all of its various repository senses are only coincidentally related to the savings institution sense.

Once we have determined that we are dealing with a polysemous lexeme, we are of course still left with the task of managing the potentially numerous polysemous senses associated with it. In particular, for any given *single* lexeme we would like to be able to answer the following questions.

- What distinct senses are there?
- How are these senses related?
- How can they be reliably distinguished?

The answers to these questions can have serious consequences for well how semantic analyzers, search engines, generators, and machine translation systems perform their respective tasks. The first two questions will be covered

here and in Section 16.4, while the final question will be covered in depth in Chapter 17.

The issue of deciding how many distinct senses should be associated with a given polysemous lexeme is a task that has long vexed lexicographers, who until recently have been the only people engaged in the creation of large lexical databases. Most lexicographers take the approach of creating entries with as many senses as necessary to account for all the fine distinctions in meaning observed in some very large corpus of examples. This is a reasonable approach given that the primary use for a traditional dictionary is to assist users in learning the various uses of a word. Unfortunately, it tends to err on the side of making more distinctions than are normally required for any reasonable computational application.

To make this notion of distinguishing distinct senses more concrete, consider the following uses of the verb *serve* from the WSJ corpus.

(16.8) They rarely *serve* red meat, preferring to prepare seafood, poultry or game birds.

(16.9) He *served* as U.S. ambassador to Norway in 1976 and 1977.

(16.10) He might have *served* his time, come out and led an upstanding life.

Reasonable arguments can be made that each of these examples represents a distinct sense of *serve*. For example, the implicit contrast between *serving red meat* and *preparing seafood* in the first example indicates a strong connection between this sense of *serve* and the related notion of food preparation. Since there is no similar component in any of the other examples, we can assume that this first use is distinct from the other two. Next, we might note that the second example has a different syntactic subcategorization from the others since its first argument, which denotes the role played by the subject, is a prepositional phrase. As will be discussed in Section 16.3, such differing syntactic behaviors are often symptomatic of differing underlying senses. Finally, the third example is specific to the domain of incarceration. This is clear since this example provides almost no specific information about prison, and yet has an obvious and clear meaning; a meaning which plays no role in the other examples.

Another practical technique, for determining if two distinct senses are present is to combine two separate uses of a lexeme into a single example using a conjunction, a device has the rather improbable name of **zeugma**. Consider the following ATIS examples.

ZEUGMA

(16.11) Which of those flights serve breakfast?

(16.12) Does Midwest express serve Philadelphia?

(16.13) ?Does Midwest express serve breakfast and Philadelphia?

The oddness of invented third example indicates there is no sensible way to make a single sense of *serve* work for both breakfast and Philadelphia. More precisely, the underlying concepts invoked by *serve* in the first example can not be applied in any meaningful way to *Philadelphia*. This is an instance where we can make use of examples from a corpus along with our native intuitions in a structured way to discover the presence or distinct senses.

WORD SENSE
DISAMBIGUA-
TION

The issue of discovering the proper set of senses for a given lexeme is distinct from the process of determining which sense of a lexeme is being used in a given example. This latter task is called **word sense disambiguation**, or **word sense tagging** by analogy to part-of-speech tagging, and is covered in detail in Chapter 17. As this analogy implies, the task typically presumes that a *fixed* set of senses can be associated with each lexical item, a dubious proposition that we will take up in Section 16.4.

Finally, let us turn briefly to the topic of relatedness among the various senses of a single polysemous lexeme. Earlier, we made an appeal to the intuition that the polysemous senses of a lexeme are unlikely to have come about by coincidence. This raises the obvious question that if they are not related by coincidence, how are they related. This question has not received much attention from those constructing large lexicons since as long as the lexicon contains the correct senses, how they came to be there is largely irrelevant. However, as soon as applications begin to deal with a wide variety of inputs, they encounter novel uses that do not correspond to any of the static senses in the system's lexicon. By examining the systematic relations among listed senses, we can gain insight into the meanings of such novel uses. These notions will be discussed in more detail in Section 16.4.

Synonymy

SYNONYMY

SUBSTI-
TUTABILITY

The phenomenon of synonymy is sufficiently widespread to account for the popularity of both thesauri and crossword puzzles. As with homonymy, the notion of **synonymy**, has a deceptively simple definition: *different lexemes with the same meaning*. Of course, this definition leaves open the question of what it means for two lexemes to mean the same thing. Although Section 16.3 will provide some answers to this question, we can make progress without answering it directly by invoking the notion of **substitutability**: two lexemes will be considered synonyms if they can substituted for one another in a sentence without changing either the meaning or the acceptability of the sentence. The following ATIS examples illustrate this notion of substi-

tutability.

(16.14) How big is that plane?

(16.15) Would I be flying on a large or small plane?

Exchanging *big* and *large* in these examples has no noticeable effect on either the meaning or acceptability of these sentences. We can take this as evidence for the synonymy of *big* and *large*, at least for these examples. Note that this is intended to be a very narrow statement. In particular, we are not saying anything about the relative likelihood of occurrence of *big* and *large* in contexts similar to these.

Not surprisingly, if we take the notion of substitutability to mean substitutable in all possible environments, then true synonyms in English are few and far between, as it is almost always possible to find some sentence where a purported synonym fails to substitute successfully. Given this, we will fall back on a weaker notion that allows us to call two lexemes synonyms if they are substitutable in *some* environment. This is, for all practical purposes, the notion of synonymy used in most dictionaries and thesauri.

The success or failure of the substitution of a given pair of candidate synonyms in a given setting depends primarily on four influences: polysemy, subtle shades of meaning, collocational constraints, and register. As we will see, only the first two involve the notion of meaning.

To explore the effect of polysemy on substitutability, consider the following WSJ example where a substitution of *large* for *big* clearly fails.

(16.16) Miss Nelson, for instance, became a kind of big sister to Mrs. Van Tassel's son, Benjamin.

(16.17) ?Miss Nelson, for instance, became a kind of large sister to Mrs. Van Tassel's son, Benjamin.

The source of this failure is the fact that the lexeme *big* has as one of its distinct polysemous senses the notion of being older, or grown up. Since the lexeme *large* lacks this sense among its many meanings, it is not substitutable for *big* in those environments where this sense is required. In this instance, the result is a sentence with a different meaning altogether. In other cases, such a substitution may result in a sentence that is either odd or entirely uninterpretable.

We referred to the next influence on synonymy as *shades of meaning*. By this, we have in mind cases where two lexemes share a central core meaning, but where additional ancillary facts are associated with one the lexemes. Consider the use of the lexemes *price* and *fare* in the ATIS corpus.

Semantically, both have the notion of the cost for a service at the core of their meanings. They are not, however, freely interchangeable. Consider the following ATIS examples.

(16.18) What is the cheapest first class fare?

(16.19) ?What is the cheapest first class price?

Exchanging *price* for *fare* in this example leads to a certain amount of oddity. The source of this oddness is hard to pin down, but *fare* seems to be better suited to the costs for various services (ie. coach, business and first class fares), while *price* seems better applied to the tickets that represent these services. Of course, a more complete account of how these lexemes are used in this domain would require a systematic analysis of a corpus of examples. The point is that although these terms share a core meaning, there are subtle meaning-related differences that influence how they can be used.

These two influences on substitutability clearly involve the meanings of the lexical items. There are, however, other influences on the success or failure of a synonym substitution that are not based on meaning in any direct way. Collocational constraints are one such influence. By a collocational constraint, we mean the kind of arbitrary associations, or attractions, between lexical items that were captured using techniques such as N-grams in Chapter 6.

Consider the following WSJ example.

(16.20) We frustrate 'em and frustrate 'em, and pretty soon they make a big mistake.

(16.21) ?We frustrate 'em and frustrate 'em, and pretty soon they make a large mistake.

As this example illustrates, there is a preference for using *big* rather than *large* when referring to mistakes of a critical or important nature. This is not due to a polysemy difference, nor does it seem to be due to any subtle shaded meaning difference between *big* and *large*. Note also, that this is clearly different than the *large sister* example in that *a large mistake* is still interpretable in the correct way; it just does not seem as natural to use *large* as *big*. Therefore, in this case, we must say that there is simply an arbitrary preference for *big* as opposed to *large* as applied to *mistakes*.

REGISTER

Finally, by **register**, we mean the social factors that surround the use of possible synonyms. Here we are referring to lexemes with essentially identical meanings that are not interchangeable in all environments due to factors such as politeness, group status, and other similar social pressures. For ex-

ample, multisyllabic lexemes with Latin or Greek origins are often used in place of shorter lexemes when a technical or academic style is desired.

As was the case with homonymy, these influences on synonymy have differing practical implications for computational applications. In Chapters 19 and 20, we will see that similarity of meaning, collocational constraints, and appropriateness of use are of great importance in natural language generation and machine translation. On the other hand, in the domains of information extraction and information retrieval, appropriateness of use is of far less consequence than the notion of identity of meaning.

Hyponymy

In our discussion of *price* and *fare*, we introduced the notion of pairs of lexemes with similar but non-identical meanings. The notion of **hyponymy** is based on a restricted class of such pairings: *pairings where one lexeme denotes a subclass of the other*. For example, the relationship between *car* and *vehicle* is one of hyponymy. Since this relation is not symmetric we will refer to the more specific lexeme as a **hyponym** of the more general one, and conversely to the more general term as a **hypernym** of the more specific one. We would therefore say that *car* is a hyponym of *vehicle*, and *vehicle* is hypernym of *car*.

HYPONYM

HYPONYM

HYPERNYM

As with synonymy, we can explore the notion of hyponymy by making use of a restricted kind of substitution. Consider the following schema.

That is a *x*. \Rightarrow That is a *y*.

If *x* is a hyponym of *y*, then in any situation where the sentence on the left is true, the newly created sentence on the right must also be true, as in the following example.

That is a *car*. \Rightarrow That is a *vehicle*.

There are a number of important differences between this kind of limited substitution and the kind of substitutions discussed with respect to synonymy. There the resulting sentence could plausibly serve as a substitute for the original sentence. Here, the new sentence is not intended to be a substitution for the original, rather it merely serves as a diagnostic test for the presence of hyponymy.

The concept of hyponymy is closely related to a number of other notions that play central roles in biology, linguistic anthropology and computer science.

The term **ontology** usually refers to an analysis of some domain, or **microworld**, into a set of distinct objects. A **taxonomy** is a particular arrange-

ONTOLOGY

TAXONOMY

ment of the elements of an ontology into a tree-like class inclusion structure. Normally, there are a set of well-formedness constraints on taxonomies that go beyond their component class inclusion relations. For example, the lexemes *hound*, *mutt*, and *puppy* are all hyponyms of *dog*, but it would be odd to construct a taxonomy from those pairs since the concepts motivating the relations is different in each case. Finally, the computer science notion of an **object hierarchy** is based the notion that objects from an ontology arranged in a taxonomy, can receive, or inherit, features from their ancestors in a taxonomy. This, of course, only makes sense when the elements in the taxonomy are in fact complex structured objects with features to be inherited.

Therefore, sets of hyponymy relations, by themselves, do not constitute an ontology, category structure, taxonymy, or object hierarchy. They have, however, proved to be useful as approximations to such structures. We will return to the topic of hyponymy in Section 16.2 when we discuss the WordNet database.

16.2 WORDNET: A DATABASE OF LEXICAL RELATIONS

The widespread use of lexical relations in linguistic, psycholinguistic, and computational research has led to a number of efforts to create large electronic databases of such relations. These efforts have, in general, followed one of two basic approaches: mining information from existing dictionaries and thesauri, and handcrafting a database from scratch. Despite the obvious advantages of reusing existing resources, WordNet, the most well-developed and widely used lexical database for English, was developed using the latter approach (Beckwith *et al.*, 1991).

WordNet consists of three separate databases, one each for nouns and verbs, and a third for adjectives and adverbs; closed class lexical items are not included in WordNet. Each of the three databases consists of a set of lexical entries corresponding to unique orthographic forms, accompanied by sets of senses associated with each form. Figure 16.1 gives some idea of the scope of the current, WordNet 1.6, release. The databases can be accessed directly with a browser (locally or over the Internet), or programmatically through the use of a set of C library functions.

In their most complete form, WordNet's sense entries consist of a set of synonyms, a dictionary-style definition, or gloss, and some example uses. Figure 16.2 shows an abbreviated version of the wordnet entry for the noun *bass*. As this entry illustrates, there are several important differences be-

tween WordNet entries and our notion of a lexeme. First, since WordNet contains no phonological information, it makes no attempt to keep separate lexemes with distinct pronunciations. For example, in this entry **bass**⁴, **bass**⁵, and **bass**⁸ all refer to the [b æ s] fish sense, while the others refer to the [b ey s] musical sense. More generally, WordNet makes no attempt to distinguish homonymy from polysemy. For example, as far as this entry is concerned, **bass**¹ bears the same relationship to **bass**² as it does to **bass**⁴. This is a conservative strategy that reflects the fact that although there are fairly reliable diagnostics for discriminating among distinct word senses, systematically organizing the resulting polysemous senses is a much more uncertain and subjective activity. Given this, the developers of WordNet have opted to simply list distinct senses, without attempting to explicitly organize them in the hierarchical manner seen in many dictionaries.

Figures 16.3 and 16.4 give a rough idea of how these senses are distributed throughout the database. The distributions are extremely skewed, with a small number of entries having a large number of senses, and a large

Category	Unique Forms	Number of Senses
Noun	94474	116317
Verb	10319	22066
Adjective	20170	29881
Adverb	4546	5677

Figure 16.1 Scope of the current WordNet 1.6 release in terms of unique entries and total number of senses for the four databases.

The noun “bass” has 8 senses in WordNet.

1. bass - (the lowest part of the musical range)
2. bass, bass part - (the lowest part in polyphonic music)
3. bass, basso - (an adult male singer with the lowest voice)
4. sea bass, bass - (flesh of lean-fleshed saltwater fish of the family Serranidae)
5. freshwater bass, bass - (any of various North American lean-fleshed freshwater fishes especially of the genus *Micropterus*)
6. bass, bass voice, basso - (the lowest adult male singing voice)
7. bass - (the member with the lowest range of a family of musical instruments)
8. bass - (nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

Figure 16.2 The WordNet 1.6 entry for the noun *bass*.

number having a single sense. Distributions like this are ubiquitous when dealing with the lexicon, and are referred to as Zipf distributions (Zipf, 1949). Note also that the degree of polysemy in the verb database is higher than in the noun database. This is consistent with the fact that there are far fewer verbs than nouns in English and their meanings are far more malleable. Finally, we should note that these polysemy distributions correlate well with actual word frequency and led the WordNet developers to use degree of polysemy as a proxy for frequency in the database.

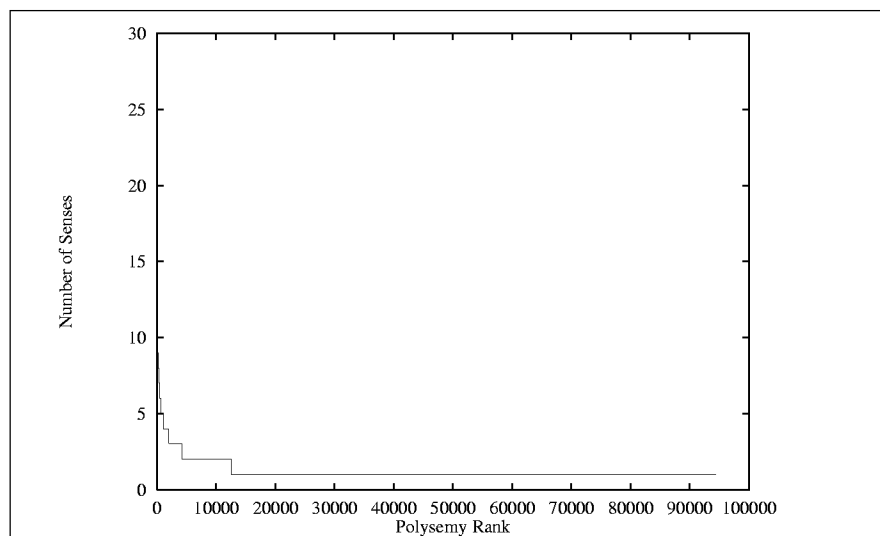


Figure 16.3 Distribution of senses among the nouns in WordNet.

Of course, a simple listing of lexical entries would not be much more useful than an ordinary dictionary. The power of WordNet lies in its set of domain-independent lexical relations. These relations can hold among WordNet entries, senses, or sets of synonyms. They are, for the most part, restricted to items with the same part-of-speech, or more pragmatically, to items within the same database. Figures 16.5, 16.6, and 16.7 show a subset of the relations associated with each of the three databases, along with a brief explanation and an example. Since a full discussion of the contents of WordNet is beyond the scope of this text, we will limit ourselves to a discussion of two of its most useful and well-developed features: its sets of synonyms, and its hyponymy relations.

The fundamental basis for synonymy in WordNet is the same as that given on page 596. Two WordNet entries are considered synonyms if they

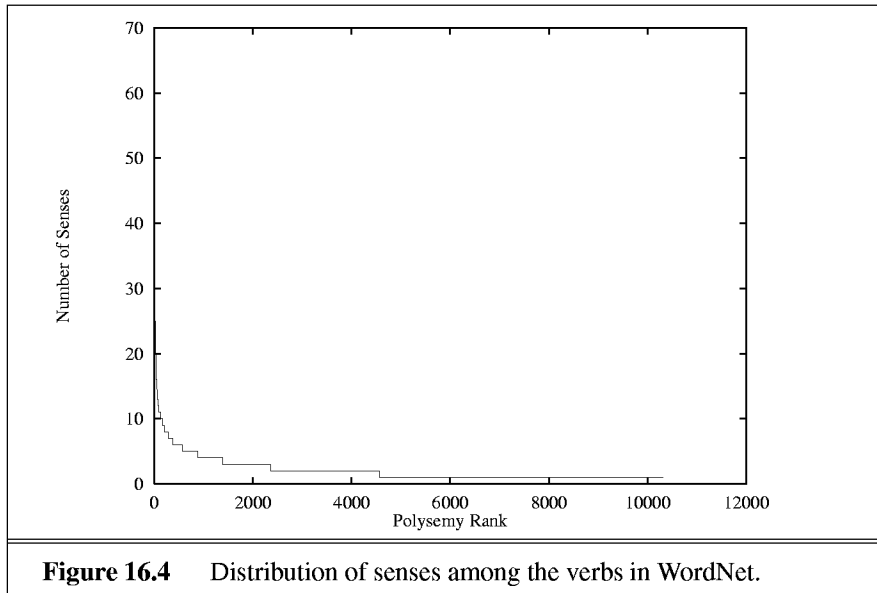


Figure 16.4 Distribution of senses among the verbs in WordNet.

Relation	Definition	Example
Hyperym	From concepts to superordinates	<i>breakfast</i> → <i>meal</i>
Hyponym	From concepts to subtypes	<i>meal</i> → <i>lunch</i>
Has-Member	From groups to their members.	<i>faculty</i> → <i>professor</i>
Member-Of	From members to their groups.	<i>copilot</i> → <i>crew</i>
Has-Stuff	From things to what they're made of.	→
Stuff-Of	From stuff to what it makes up.	→
Has-Part	From wholes to parts	<i>table</i> → <i>leg</i>
Part-Of	From parts to wholes.	<i>course</i> → <i>meal</i>
Antonym	Opposites	<i>leader</i> → <i>follower</i>

Figure 16.5 Noun Relations in WordNet.

Relation	Definition	Example
Hypernym	From events to superordinate events	<i>fly</i> → <i>travel</i>
Troponym	From events to their subtypes	<i>walk</i> → <i>stroll</i>
Entails	From events to the events they entail	<i>snore</i> → <i>sleep</i>
Antonym	Opposites	<i>increase</i> ⇔ <i>decrease</i>

Figure 16.6 Verb Relations in WordNet.

Relation	Definition	Example
Antonym	Opposite	<i>heavy</i> \longleftrightarrow <i>light</i>
Adverb	Opposite	<i>quickly</i> \longleftrightarrow <i>slowly</i>

Figure 16.7 Adjective and Adverb Relations in WordNet.

SYNSET

can be successfully substituted in some context. The particular theory and implementation of synonymy in WordNet is organized around the notion of a **synset**, a set of synonyms. Consider the following example of a synset.

{chump, fish, fool, gull, mark, patsy, fall guy, sucker, schlemiel, shlemiel, soft touch, mug}

The dictionary-like definition, or gloss, of this synset describes it as *a person who is gullible and easy to take advantage of*. Each of the lexical entries included in the synset can, therefore, be used to express this notion in some setting. In practice, synsets like this one actually *constitute* the senses associated with many WordNet entries. Specifically, it is this exact synset, with its associated definition and examples, that makes up one of the senses for each of the entries listed in the synset.

Looking at this from a more theoretical perspective, each synset can be taken to represent a concept that has become lexicalized in the language. Synsets are thus somewhat analogous to the kinds of concepts we discussed in Chapter 14. Instead of representing concepts using logical terms, WordNet represents them as lists comprised of the lexical entries that can be used to express the concept. This perspective motivates the fact that it is synsets, not lexical entries or individual senses, that participate in most of the semantic relations shown in Figures 16.5, 16.6, and 16.7.

The hyponymy relations in WordNet correspond directly to the notion of immediate hyponymy discussed on page 599. Each synset is related to its immediately more general and more specific synsets via direct hypernym and hyponym relations. To find chains of more general or more specific synsets, one can simply follow a transitive chain of hypernym and hyponym relations. To make this concrete, consider the hypernym chains for **bass**³ and **bass**⁷ shown in Figure 16.8.

In this depiction of hyponymy, successively more general synsets are shown on successive indented lines. The first chain starts from the concept of a human bass singer. Its immediate superordinate is a synset corresponding to the generic notion of a singer. Following this chain leads eventually to notions such as entertainer and person. The second chain, which starts from the musical instrument notion, has a completely different chain leading

```

Sense 3
bass, basso --
(an adult male singer with the lowest voice)
=> singer, vocalist
    => musician, instrumentalist, player
        => performer, performing artist
            => entertainer
                => person, individual, someone...
                    => life form, organism, being...
                        => entity, something
                            => causal agent, cause, causal agency
                                => entity, something

Sense 7
bass --
(the member with the lowest range of a family of
musical instruments)
=> musical instrument
    => instrument
        => device
            => instrumentality, instrumentation
                => artifact, artefact
                    => object, physical object
                        => entity, something

```

Figure 16.8 Hyponymy chains for two separate senses of the lexeme *bass*. Note that the chains are completely distinct, only converging at *entity*.

eventually such concepts as musical instrument, device and physical object. Both paths do eventually join at the synset *entity* which basically serves as a placeholder at the top of the hierarchy.

16.3 THE INTERNAL STRUCTURE OF WORDS

The approach to meaning spelled out in the last two chapters hinged on the notion that there is a fundamental predicate-argument structure underlying our meaning representations. In composing such representations, we assumed that certain classes of lexemes tend to contribute the predicate and predicate-argument structure, while others contribute the arguments. This section explores in more detail the systematic ways that the meanings of lex-

emes are structured to support this notion. In particular, it explores the notion that the meaning representations associated with lexemes have analyzable internal structures, and that it is these structures, combined with a grammar, that determine the relations among lexemes in well-formed sentences.

Thematic Roles

Thematic roles, first proposed by Gruber (1965a) and Fillmore (1968)¹ are a set of categories which provide a shallow semantic language for characterizing certain arguments of verbs. For example consider the following two WSJ fragments:

(16.22) Houston's Billy Hatcher broke a bat.

(16.23) He opened a drawer.

In the predicate calculus event representation of Chapter 14, part of the representation of these two sentences would be the following:

$$\begin{aligned} &\exists e, x, y \text{ Isa}(e, \text{Breaking}) \wedge \text{Breaker}(e, \text{BillyHatcher}) \\ &\quad \wedge \text{BrokenThing}(e, y) \wedge \text{Isa}(y, \text{BaseballBat}) \\ &\exists e, x, y \text{ Isa}(e, \text{Opening}) \wedge \text{Opener}(e, \text{he}) \\ &\quad \wedge \text{OpenedThing}(e, y) \wedge \text{Isa}(y, \text{Door}) \end{aligned}$$

DEEP ROLES

THEMATIC
ROLE

In this representation, the roles of the subjects of the verbs *break* and *open* are *Breaker* and *Opener* respectively. These **deep roles** are specific to each possible kind of event; *Breaking* events have *Breakers*, *Opening* events have *Openers*, *Eating* events have *Eaters*, and so on. But *Breakers* and *Openers* have something in common. They are both volitional actors, often animate, and they have direct causal responsibility for their events. A **thematic role** is a way of expressing this commonality. We say that the subjects of both these verbs are AGENTS. Thus AGENT is the thematic role which represents an abstract idea such as volitional causation. Similarly, the direct objects of both these verbs, the *BrokenThing* and *OpenedThing*, are both prototypically inanimate objects which are affected in some way by the action. The thematic role for these participants is the THEME.

As we will discuss below, while there is no standard set of thematic roles, there are many roles that are commonly used by computational systems. For example, in any straightforward interpretation of Example 16.24, Mr. Cockwell has had his collarbone broken, but there is no implication that he was the AGENT of this unfortunate event. This kind of participant

¹ Fillmore actually called them *deep cases*, on the metaphor of morphological case.

can be labeled an EXPERIENCER, while the directly effected participant, the collarbone in this case, is again assigned the THEME role.

- (16.24) A company soccer game last year got so rough that Mr. Cockwell broke his collarbone and an associate broke an ankle.

In Example 16.25, the earthquake is the direct cause of the glass breaking and hence might seem to be a candidate for an AGENT role. This seems odd, however, since earthquakes are not the kind of participant that can intentionally do anything. Examples such as this have been the source of considerable debate over the years among the proponents of various thematic role theories. Two approaches are common: assign the earthquake to the AGENT role and assume that the intended meaning has some kind of metaphorical connection to the core animate/volitional meaning of AGENT, or add a role called FORCE that is similar to AGENT but lacks any notion of volitionality. We will follow this latter approach and return to the notion of metaphor in Section 16.4.

- (16.25) The quake broke glass in several downtown skyscrapers.

Finally, in Example 16.26, the subject (*it*) refers to an event participant (in this case, someone else's elbow) whose role in the breaking event is as the instrument of some other agent or force. Such participants are called INSTRUMENTS.

- (16.26) It broke his jaw.

Figure 16.9 presents a small list of commonly-used thematic roles along with a rough description of the meaning of each. Figure 16.10 provides representative examples of each of role. Note that this list of roles is by no means definitive, and does not correspond to any single theory of thematic roles.

Applications to Linking Theory and Shallow Semantic Interpretations

One common use thematic roles in computational systems is as a shallow semantic language. For example, as Chapter 21 will describe, thematic roles are sometimes used in machine translation systems as part of a useful intermediate language.

Another use of thematic roles, which was part of their original motivation in Fillmore (1968), was as an intermediary between semantic roles in conceptual structure or common-sense knowledge like *Breaker* and *Driven-Thing* and their more language-specific surface grammatical realization as

Thematic Role	Definition
AGENT	The volitional causer of an event
EXPERIENCER	The experiencer of an event
FORCE	The non-volitional causer of the event
THEME	The participant most directly affected by an event
RESULT	The end product of an event
INSTRUMENT	An instrument used in an event
BENEFICIARY	The beneficiary of an event
SOURCE	The origin of the object of a transfer event
GOAL	The destination of an object of a transfer event
Figure 16.9 Some commonly-used thematic roles with their definitions.	

Thematic Role	Example
AGENT	<i>The waiter</i> spilled the soup
EXPERIENCER	<i>John</i> has a headache
FORCE	<i>The wind</i> blows debris from the mall into our yards
THEME	Only after Benjamin Franklin broke <i>the ice</i> ...
RESULT	The French government has built a <i>regulation-size baseball diamond</i> ...
INSTRUMENT	He turned to poaching catfish, stunning them <i>with a shocking device</i>
BENEFICIARY	Whenever Ann Callahan makes hotel reservations <i>for her boss</i> ...
SOURCE	I flew in <i>from Boston</i> .
GOAL	I drove <i>to Portland</i> .
Figure 16.10 Prototypical examples of various thematic roles.	

subject and object. Fillmore noted that there are prototypical patterns governing which argument of a verb will become the subject of an active sentence, proposing the following hierarchy (often now called a **thematic hierarchy** (Jackendoff, 1972)) for assigning the subject role:

AGENT \succ INSTRUMENT \succ THEME

Thus if the thematic description of a verb includes an AGENT, an INSTRUMENT, and a THEME, it is the AGENT which will be realized as the subject. If the thematic description only includes an INSTRUMENT and a THEME, it is the INSTRUMENT which will become the subject. The thematic hierarchy is used in reverse for determining the direct object of active sentences, or the subject of passive sentences. Here are examples from Fillmore

(1968) using the verb *open*:

(16.27) *John opened the door.*

AGENT THEME

(16.28) *John opened the door with the key.*

AGENT THEME INSTRUMENT

(16.29) *The key opened the door.*

AGENT THEME

(16.30) *The door was opened by John.*

THEME AGENT

This approach led to a wide variety of work over the last thirty years on the mapping between conceptual structure and grammatical function, in an area generally referred to as **linking theory**. For example many scholars such as Talmy (1985), Jackendoff (1983b), and Levin (1993) show that semantic properties of verbs help predict which surface **alternations** they can take. An alternation is a set of different mappings of conceptual (deep) roles to grammatical function. For example Fillmore (1965) and very many subsequent researchers have studied the **dative alternation**, the phenomenon that certain verbs like *give*, *send*, or *read* which can take an AGENT, a THEME, and a GOAL, allow the THEME to appear as object and the GOAL in a prepositional phrase (as in 16.31a), or the GOAL to appear as the object, and the THEME as a sort of ‘second object’ (as in 16.31b):

LINKING
THEORY

ALTERNATIONS

DATIVE
ALTERNATION

(16.31) a. *Doris gave/sent/read the book to Cary.*

AGENT THEME GOAL

b. *Doris gave/sent/read Cary the book.*

AGENT GOAL THEME

Many scholars, including Green (1974), Pinker (1989), Gropen *et al.* (1989), Goldberg (1995) and Levin (1993) (see Levin (1993, p. 45) for a full bibliography), have argued this alternation occurs with particular semantic classes of verbs, including (from Levin) ‘verbs of future having’ (*advance*, *allocate*, *offer*, *owe*), ‘send verbs’ (*forward*, *hand*, *mail*), ‘verbs of throwing’ (*kick*, *pass*, *throw*, and many other classes).

Similarly, Talmy (1985), following Lakoff (1965, p.126), shows that ‘affect’ verbs such as *frighten*, *please*, and *exasperate* can appear with the THEME as subject, as in (16.32), or with the EXPERIENCER as subject and the THEME as a prepositional object, as in (16.33):

(16.32) a. *That frightens me.*

THEME EXPERIENCER

b. *That interests me.*

THEME EXPERIENCER

- c. *That surprises me.*
 THEME AGENT
- (16.33) a. *I am frightened of that.*
 EXPERIENCER THEME
- b. *I am interested in that.*
 EXPERIENCER THEME
- c. *I am surprised at that.*
 EXPERIENCER THEME

Levin (1993) summarizes 80 of these alternations, including extensive lists of the verbs in each semantic class, together with the semantic constraints, exceptions, and other idiosyncracies. This list has been used in a number of computational models (e.g. Dang *et al.*, 1998; Jing and McKeown, 1998)

While research of the type summarized above has shown a relation between verbal semantic and syntactic realization, it is less clear that this relation is mediated by a small set of thematic roles, with or without a thematic hierarchy. For example, it turns out that semantic classes are insufficient to define the set of verbs that participate in an alternation. For example many verbs do not allow the dative alternation despite being in the proper semantic class (e.g. *donate*, *return*, *transfer*). In addition, as shown above, many of the verbal alternations violate any standard thematic hierarchy (dative alternation sentences like *Ling sent Mary the book* have a GOAL as direct object followed by an oblique THEME, when THEME should be the best direct object). Furthermore, arguments about the appropriate set of thematic roles are legion. But an even greater problem is that thematic roles, however they are defined, could only play a very small role in the general mapping from semantics to syntax. This is because thematic roles are only relevant to determining the grammatical role of NP and PP arguments, and play no part in the realization of other arguments of verbs and other predicates. Many such possible arguments were described in Figure 11.3 on page 411, such as sentential complements (**Sfin**, **Swh-**, **Sforto**), verb phrases (**VPbrst**, **VPto**, etc), or quotations (**Quo**). Furthermore, thematic roles only are useful in mapping the arguments of verbs; but nouns, for example, have arguments as well (*destruction of the city*, *father of the bride*).

There are a number of possible responses to these problems with thematic roles. Many systems continue to use them for such practical purposes as interlinguas in machine translation or as a convenient level of shallow semantic interpretation. Other researchers have argued that thematic roles should be considered an epiphenomenon, rather than a distinct represen-

tational level. For example following Foley and van Valin (1984), Dowty (1991) argues that rather than a discrete set of thematic roles there are only two cluster-concepts, PROTO-AGENT and PROTO-PATIENT. Determining whether an argument of a verb is a PROTO-AGENT is predictable from the entailments of the deep conceptual structure meaning of the verb. The mapping from semantic role in conceptual structure to grammatical function proceeds via simple rules (the most PROTO-AGENT-like of the arguments is the subject, the most PROTO-PATIENT-like is the object (or the subject of the passive construction)). Dowty's two rules make direct reference to the deep conceptual structure of the verb; thus thematic roles do not appear at any representational level at all.

One problem with Dowty's model is that the choice of thematic roles is not always predictable from the underlying conceptual structure of the event and its participants. For example Fillmore (1977) pointed out that the different verbs which can describe a **commercial event** each choose a different way to map the participants of the event. For example, a transaction between Amie and Benson involving three dollars and a sandwich can be described in any of these ways:

- (16.34) a. Amie bought the sandwich from Benson for three dollars.
b. Benson sold Amie the sandwich for three dollars.
c. Amie paid Benson three dollars for the sandwich.

Each of these verbs *buy*, *sell*, and *pay*, chooses a different **perspective** on the commercial event, and realizes this perspective by choosing a different mapping of underlying participants to thematic roles. The fact that these three verbs have very different mappings suggests that the thematic roles for a verb must be listed in the lexical entry for the verb, and are not predictable from the underlying conceptual structure.

This fact, together with the fact mentioned earlier that verb alternations are not completely predictable semantically (e.g. exceptions like *donate*) has led many researchers to assume that any useful computational lexicon needs to list for each verb (or adjective or other predicate) its syntactic and thematic combinatory possibilities. Another advantage of listing the combinatory possibilities for each verb is that the probability of each thematic frame can also be listed.

One recent attempt to list these elements for a number of predicates of English is the FRAMENET project (Baker *et al.*, 1998; Lowe *et al.*, 1997). A FRAMENET entry for a word lists every set of arguments it can take, including the possible sets of thematic roles, syntactic phrases, and their grammat-

ical function. The thematic roles used in FRAMENET are much more specific than the 9 examples we've been describing. Each FRAMENET thematic role is defined as part of a **frame**, and each **frame** as part of a domain. For example the **Cognition** domain has frames like **static cognition** (*believe, think, understand, etc.*), **cogitation** (*brood, ruminate*), **judgment**, (*accuse, admire, rebuke*), etc. All of the cognition frames define the thematic role COGNIZER. In the **judgment** frame, the COGNIZER is referred to as the JUDGE; the frame also includes an EVALUEE, a REASON, and a ROLE; here are some examples from (Johnson, 1998):

Judge	Kim respects Pat for being so brave
Evaluee	Kim respects Pat for being so brave
Reason	Kim respects Pat for being so brave
Role	Kim respects Pat as a scholar

Each entry is also labeled by one of the **phrase types** described in Figure 11.3 on page 411, and by a grammatical function (subject, object, or complement). For example, here is part of the FRAMENET entry for the judgment verb *appreciate*; we have shown only the active senses of the verb; the full entry includes passives as well. Example sentences are (sometimes shortened) from the British National Corpus:

- (16.35)
- | | | | | | |
|----|---------|--------------------|---------------|------------------------|-------------|
| a. | JUDGE | | REASON | | EVALUEE |
| | NP/Subj | | NP/Obj | | PP(in)/Comp |
| | I | still appreciate | good manners | in men. | |
| b. | JUDGE | | EVALUEE | REASON | |
| | NP/Subj | | NP/Obj | PP(for)/Comp | |
| | I | could appreciate | it | for the music alone. | |
| c. | JUDGE | | REASON | | |
| | NP/Subj | | NP/Obj | | |
| | I | appreciate | your kindness | | |
| d. | JUDGE | | EVALUEE | ROLE | |
| | NP/Subj | | NP/Obj | PP(for)/Comp | |
| | He | did not appreciate | the artist | as a dissenting voice. | |

By contrast, another sense of the verb *appreciate* is as a verb of static cognition like *understand*; verbs of static cognition have roles like COGNIZER and CONTENT; here are some examples:

- (16.36)
- | | | | |
|----|----------|-------------|--|
| a. | COGNIZER | | CONTENT |
| | NP/Subj | | Sfin/Comp |
| | They | appreciate | that communication is a two-way process. |
| b. | COGNIZER | | CONTENT |
| | NP/Subj | | Swh-/Comp |
| | She | appreciated | how far she had fallen from grace. |

It should be clear from examining the example sentences that some generalizations can be drawn about the realization of different thematic roles. JUDGES, COGNIZERS, and AGENTS in general are often realized as subjects of active sentences. ROLES are often realized as PPs with the preposition *as*. CONTENT is often realized as some kind of S. Representing thematic roles at this fine-grained level may thus make the mapping to syntax more transparent. The problem with a scheme like FRAMENET is the extensive human effort it requires in defining thematic roles for each domain and each frame.

Selection Restrictions

The notion of a **selection restriction** can be used to augment thematic roles by allowing lexemes to place certain semantic restrictions on the lexemes and phrases that can accompany them in a sentence. More specifically, a selection restriction is a semantic constraint imposed by a lexeme on the concepts that can fill the various argument roles associated with it. As with many other kinds of linguistic constraints, selection restrictions can most easily be observed in situations where they are violated. Consider the following example originally discussed in Chapter 14.

SELECTION
RESTRICTION

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

There are two possible parses for this sentence corresponding to the intransitive and transitive versions of the verb *eat*. These two parses lead, in turn, to two distinct semantic analyses. In the intransitive case, the phrase *someplace that's close to ICSI* is an adjunct that modifies the event specified by the verb phrase, while in the transitive case it provides a true argument to the eating event. This latter case is similar in structure and interpretation to examples such as the following, where the noun phrase specifies the thing to be eaten.

(16.38) I wanna eat some really cheap Chinese food right now.

Not surprisingly, attempting to analyze Example 16.37 along these lines results in a kind of semantic ill-formedness. This ill-formedness signals the presence of a selection restriction imposed by *eat* on its PATIENT role: it has to be something that is edible. Since the phrase being proposed as the PATIENT in this scenario can not easily be interpreted as edible, the interpretation exhibits the semantic analog of syntactic ungrammaticality. This particular variety of ill-formedness arises from what is known as a **selection restriction violation**: a situation where the semantics of the filler of a

SELECTION
RESTRICTION
VIOLATION

thematic role is not consistent with a constraint imposed on the role by the predicate.

This rather informal description of selection restrictions needs to be refined in a number of ways before it can be put to practical use. The first refinement concerns the proper locus for stating the selection restrictions. As discussed in Section 16.1, lexemes are often associated with a wide variety of different senses and, not surprisingly, these senses can enforce differing constraints on their arguments. Selection restrictions therefore are associated with particular senses, not entire lexemes. Consider the following examples of the lexeme *serve*.

(16.39) Well, there was the time they served green-lipped mussels from New Zealand.

(16.40) Which airlines serve Denver?

(16.41) Which ones serve breakfast?

Example 16.39 illustrates the cooking sense of *serve*, which ordinarily restricts its PATIENT to be some kind of foodstuff. Example 16.40 illustrates the *provides a commercial service to* sense of *serve*, which constrains its PATIENT to be some type of identifiable geographic or political entity. The sense shown in the third example is closely related to the first, and illustrates a sense of *serve* that is restricted to specifications of particular meals. These differing restrictions on the same thematic role of a polysemous lexeme can be accommodated by associating them with distinct senses of the same lexeme. As we will discuss in Chapter 17, this strongly suggests that selection restrictions can be used to discriminate these senses in context.

Note that the selection restrictions imposed by different lexemes, and different senses of the same lexeme, may occur at widely varying levels of specificity, with some lexemes expressing very general conceptual categories, and others expressing very specific ones indeed. Consider the following examples of the verbs *imagine*, *lift* and *diagonalize*.

(16.42) In rehearsal, I often ask the musicians to imagine a tennis game.

(16.43) Others tell of jumping over beds and couches they can't imagine clearing while awake.

(16.44) I cannot even imagine what this lady does all day.

(16.45) Atlantis lifted Galileo from the launch pad at 12:54 p.m. EDT and released the craft from its cargo bay about six hours later.

(16.46) When the battle was over, Mr. Kruger lifted the fish from the water, gently removed the hook from its jaw, admired it, and eased it back into the lake.

(16.47) To diagonalize a matrix, is to find its eigenvalues.

Given the meaning of *imagine*, it is not surprising to find that it places few semantic restrictions on the concepts that can fill its PATIENT role. Its AGENT role, on the other hand, is restricted to humans and other animate entities. In contrast, the sense of *lift* shown in Examples 16.45 and 16.46 limits its PATIENT to be something liftable, which as these examples illustrate is a notion that must cover both spacecraft and fish. For all practical purposes, this notion is best captured by the fairly general notion such as *physical object*. Finally, we have *diagonalize* which imposes a very specific constraint on the filler of its PATIENT role: it has to be a matrix.

These examples serve to illustrate an important fact about selection restrictions: the concepts, categories, and features that are deployed by the lexicon as selection restrictions are not a part of the finite language capacity. Rather, they are as open-ended as the lexicon itself. This distinguishes selection restrictions from some of the other finite features of language that are used to define lexemes including parts-of-speech, thematic roles, and semantic primitives.

Before we move on, it is worth pointing out that verbs are not the only part-of-speech that can impose selection restrictions on their arguments. Rather, it appears to be the case that any predicate-bearing lexeme can impose arbitrary semantic constraints on the concepts that fill its argument roles. Consider the following examples, which illustrate the selection restrictions associated with some non-verb parts-of-speech.

(16.48) Radon is a naturally occurring odorless, tasteless gas that can't be detected by human senses.

(16.49) What is the lowest fare for United Airlines flight four thirty?

(16.50) Are there any restaurants open after midnight?

The adjectives *odorless* and *tasteless* in 16.48 are restricted to concepts that can possess an odor or a taste. Similarly, as we discussed earlier in Section 16.1, the noun *fare* is restricted to various forms of public transportation. Finally, arguments to the preposition *after* must directly or indirectly designate points in time.

Representing Selection Restrictions

The semantics of selection restrictions can be captured in a straightforward way by extending the event-oriented meaning representations employed in Chapter 14. Recall that the representation of an event consists of a single variable that stands for the event, a predicate that denotes the kind of event, and a series of variables and relations that designate the roles associated with the event. Ignoring the issue of the λ -structures, and using thematic roles rather than deep event roles, the semantic contribution of a verb like *eat* might look like the following.

$$\exists e, x, y \text{ Eating}(e) \wedge \text{Agent}(e, x) \wedge \text{Patient}(e, y)$$

With this representation, all we know about y , the filler of the *Patient* role, is that it is associated with an *Eating* event via the *Patient* relation. To stipulate the selection restriction that y must be something edible, we simply add a new term to that effect, as in the following.

$$\exists e, x, y \text{ Eating}(e) \wedge \text{Eater}(e, x) \wedge \text{Patient}(e, y) \wedge \text{Isa}(y, \text{EdibleThing})$$

When a phrase like *ate a hamburger* is encountered, a semantic analyzer can form the following kind of representation.

$$\exists e, x, y \text{ Eating}(e) \wedge \text{Eater}(e, x) \wedge \text{Patient}(e, y) \wedge \text{Isa}(y, \text{EdibleThing}) \\ \wedge \text{Isa}(y, \text{Hamburger})$$

This representation is perfectly reasonable since the membership of y in the category *Hamburger* is consistent with its membership in the category *EdibleThing*, assuming a reasonable set of facts in the knowledge base. Correspondingly, the representation for a phrase such as *ate a takeoff* would be ill-formed because membership in an event-like category such as *Takeoff* would be inconsistent with membership in the category *EdibleThing*.

While this approach adequately captures the semantics of selection restrictions, there are two practical problems with its direct use. First, using the full power of First Order Logic to perform the simple task of enforcing selection restrictions is overkill. There are far simpler formalisms that can do the job with far less computational cost. The second problem is that it presupposes a large logical knowledge-base of facts about the concepts that make up selection restrictions. Unfortunately, although such common sense knowledge-bases are being developed, none are widely available and few have the kind of scope necessary to the task.

A far more practical approach, at least for English, is to exploit the hyponymy relations present in the WordNet database. In this approach, selection restrictions on semantic roles are stated in terms of WordNet synsets,

```

Sense 1
hamburger, beefburger --
(a fried cake of minced beef served on a bun)
=> sandwich
    => snack food
        => dish
            => nutriment, nourishment, sustenance...
                => food, nutrient
                    => substance, matter
                        => object, physical object
                            => entity, something

```

Figure 16.11 Evidence from WordNet that hamburgers are edible.

rather than logical concepts. A given meaning representation can be judged to be well-formed if the lexeme that fills a thematic role has as one of its hypernyms, the synset specified by the predicate for that thematic role. Consider how this approach would work with our *ate a hamburger* example. Among its 60,000 synsets, WordNet includes the following one, which is glossed as *any substance that can be metabolized by an organism to give energy and build tissue*.

{food, nutrient}

Given this synset, we can specify it as the selection restriction on the PATIENT role of the verb *eat*, thus limiting fillers of this role to lexemes in this synset *and its hyponyms*. Luckily, the chain of hypernyms for *hamburger* shown in Figure 16.3, reveals that that hamburgers are indeed food.

Note that in this approach, the filler of a role does not have to match the restriction synset exactly. Rather, a selection restriction is satisfied if the filler has the restricting synset as one of its eventual hypernyms. Thus in the hamburger example, the selection restriction synset is found five hypernym levels up from *hamburger*.

Of course, this approach also allows individual lexemes to satisfy restrictions at varying levels of specificity. For example, consider what happens when we apply this approach to the PATIENT roles of the verbs *imagine*, *lift* and *diagonalize*, discussed earlier. Let us restrict *imagine*'s PATIENT to the synset {entity, something}, *lift*'s PATIENT to {object, physical object} and *diagonalize* to {matrix}. This arrangement correctly permits *imagine a hamburger* and *lift a hamburger*, while also correctly ruling out *diagonalize a hamburger*.

Note that this approach relies on the presence in WordNet of exactly those lexemes that specify exactly the concepts needed for all possible selection restrictions. Unfortunately, there is no particular reason to believe that the set of concepts used as selection restrictions in a language is exactly subsumed by the lexemes in the language. This situation is accommodated to some extent in WordNet through the use of collocations such as *physical object* and *snack food*.

To address this problem more directly, there are a number of linguistically-oriented taxonomies that sit somewhere between common sense knowledge-bases such as CYC, and lexical databases such WordNet. The objects contained in these hybrid models do not have to correspond to individual lexical items, but rather to those concepts that are known to be grammatically and lexically relevant. In most cases, the upper portions of these taxonomies are taken to represent domain and language-independent notions, such as physical objects, states, events and animacy. One of the most well-developed of these ontologies is the the PENMAN Upper Model, discussed in more detail in Chapter 20.

Primitive Decomposition

The theories of meaning representation presented here, and in the last few chapters, have had a decidedly lexical flavor. The meaning representations for sentences have been composed of atomic symbols that appear to correspond very closely to individual lexemes. However, other than thematic roles, these lexical representations have had not much of an internal structure. The notion of **primitive decomposition**, or **componential analysis**, is an attempt to supply such a structure.

To explore these notions, consider the following examples motivated by the discussion in McCawley (1968).

(16.51) Jim killed his philodendren.

(16.52) Jim did something to cause his philodendren to become not alive.

One can make an argument that these two sentences mean the same thing. However, this is not case of synonymy, since *kill* is not synonymous with any individual lexemes in 16.52. Instead, one can think of *kill* as being equivalent to the particular configuration of *more fundamental* elements found in the second sentence.

Taking this to the next logical step, we can invoke the notion of canonical form and say that these two examples should have the *same* meaning

representation — the one underlying Example 16.52. Translating a simple predicate like *kill* into a more complex set of predicates can be viewed as breaking down, or decomposing, the meaning of words into combinations of simpler, more primitive, parts. In this example, the more primitive, possibly atomic, parts are the meaning representations associated with the lexemes *cause*, *become not*, and *alive*.

While many such primitive sets of have been proposed, the approach known as Conceptual Dependency (CD) (Schank, 1972) has been the most widely used primitive-based representational system within natural language processing. In this approach, eleven primitive predicates are used to represent all predicate-like language expressions. Figure 16.12 shows the eleven primitives with a brief explanation of their meaning.

As an example of this approach, consider the following sentence along with its CD representation.

(16.53) The waiter brought Mary the check.

$$\begin{aligned} \exists x, y \text{ Atrans}(x) \wedge \text{Actor}(x, \text{Waiter}) \wedge \text{Object}(x, \text{Check}) \wedge \text{To}(x, \text{Mary}) \\ \wedge \text{Ptrans}(y) \wedge \text{Actor}(y, \text{Waiter}) \wedge \text{Object}(y, \text{Check}) \wedge \text{To}(y, \text{Mary}) \end{aligned}$$

Here, the verb *brought* is translated into the two primitives *ATRANS* and *PTRANS* to indicate the fact that the waiter both physically conveyed the check to Mary and passed control of it to her. Note that CD also associates a fixed set of thematic roles with each primitive to represent the various participants in the action.

Note that, in general, the compositional approach need not be limited to the meanings of verbs. The same notion can be used to decompose nominals into more primitive notions. Consider the following decompositions of the lexemes *kitten*, *puppy*, and *child* into more primitive elements.

$$\begin{aligned} \exists x \text{ Isa}(x, \text{Feline}) \wedge \text{Isa}(x, \text{Youth}) \\ \exists x \text{ Isa}(x, \text{Canine}) \wedge \text{Isa}(x, \text{Youth}) \\ \exists x \text{ Isa}(x, \text{Human}) \wedge \text{Isa}(x, \text{Youth}) \end{aligned}$$

Here the primitives represent more primitive categories of objects, rather than actions. Using these primitives, the close relationship between these lexemes and the related terms *cat*, *dog* and *person* can then be captured with the following similar formulas.

$$\begin{aligned} \exists x \text{ Isa}(x, \text{Feline}) \wedge \text{Isa}(x, \text{Adult}) \\ \exists x \text{ Isa}(x, \text{Canine}) \wedge \text{Isa}(x, \text{Adult}) \\ \exists x \text{ Isa}(x, \text{Human}) \wedge \text{Isa}(x, \text{Adult}) \end{aligned}$$

The primary applications of primitives in natural language processing have been in semantic analysis and in machine translation. In semantic anal-

Primitive	Definition
ATRANS	The abstract transfer of possession or control from one entity to another.
PTRANS	The physical transfer of an object from one location to another
MTRANS	The transfer of mental concepts between entities or within an entity.
MBUILD	The creation of new information within an entity.
PROPEL	The application of physical force to move an object.
MOVE	The integral movement of a body part by an animal.
INGEST	The taking in of a substance by an animal.
EXPEL	The expulsion of something from an animal.
SPEAK	The action of producing a sound.
ATTEND	The action of focusing a sense organ.
Figure 16.12 A set of conceptual dependency primitives	

ysis, the principle use has been in organizing the inference process. Instead of having to encode thousands of idiosyncratic meaning postulates with particular lexical items, inference rules can be associated with a small number of primitives. We should note the use of primitive decomposition in the representation on nominals has largely been supplanted by the use of inheritance hierarchies. As we will see in Chapter 21, the emphasis in machine translation has been on the use of primitives as language independent meaning representations, or **interlinguas**.

Semantic Fields

The lexical relations described in Section 16.1 had a decidedly local character, and made no use of the internal structure of the lexemes taking part in the relation. The notion of a **semantic field** is an attempt to capture a more integrated, or wholistic, relationship among entire sets of words from a single domain. Consider the following set of words extracted from the ATIS corpus.

reservation, flight, travel, buy, price, cost, fare, rates, meal, plane

It is certainly possible to assert individual lexical relations between many of the lexemes in this list. The resulting set of relations does not, however, add up to a complete account of how these lexemes are related. They are clearly all defined with respect to a coherent chunk of common sense

background information concerning air travel. Background knowledge of this kind has been studied under a variety of frameworks and is known variously as a frame (Fillmore, 1985), model (Johnson-Laird, 1983), or script (Schank and Abelson, 1977), and plays a central role in a number of computational frameworks, some of which will be discussed in Chapter 18.

The **FrameNet** project (Baker *et al.*, 1998) is a recent attempt to provide a robust resource for this kind of knowledge. In FrameNet, lexemes that refer to actions, events, thematic roles, and objects belonging to a particular domain are linked to concepts contained in frames that represent that particular domain. As in most current ontology efforts, these frames are arranged in a hierarchy so that specific frames can inherit roles from more abstract frames. The current FrameNet effort is directed at the creation of several thousand frame-semantic lexical entries. The domains to be covered include: HEALTH CARE, CHANCE, PERCEPTION, COMMUNICATION, TRANSACTION, TIME, SPACE, BODY, MOTION, LIFE STAGES, SOCIAL CONTEXT, and COGNITION.

FRAMENET

16.4 CREATIVITY AND THE LEXICON

The approach we have presented thus far views the lexicon as a static repository from which meaning representations are retrieved as needed. A more realistic alternative view holds that the lexicon is closer to a generative device than a static repository. Rather than simply retrieving static senses, the lexicon *generates* meaning components appropriate to each situation on demand. Under this view, much of the apparent polysemy in the lexicon is due to this generative capacity. This capacity is, of course, not unlimited or unsystematic. Rather, it is governed by a number of productive, or generative, **models** that can systematically combine lexical, grammatical, contextual, and common sense knowledge to create the novel meanings we see every day.

To make this discussion more concrete, consider the following sentence from the WSJ corpus.

(16.54) That doesn't scare Digital, which has grown to be the world's second-largest computer maker by poaching customers of IBM's mid-range machines.

Let's consider the meanings of *scare* and *poach* in this example. The verb *scare* in WordNet has two closely related senses: to cause fear in, and to

cause to lose courage. Although it might be interesting to consider which of these senses is the right one for this example, it's even more interesting to consider what it would mean for a corporation to lose courage, or even to have it in the first place. For this sentence to make sense, it would appear to be the case that corporations must be able to experience emotions like fear or courage. Of course, they don't but we certainly speak of them and often reason about them as if they do.

The verb *poach* in WordNet has a *cooking by boiling* sense, and a *illegal taking of game* sense. Intuitively, the use of *poach* in this example is closer to the illegal taking meaning than the boiling one. Of course, this is clearly not a simple instance of this use; the poaching involved is not illegal, and we can only hope that the poached things are not being killed. In this case, the customers are being viewed as a kind of property belonging to the company they do business with; and when they choose to do business with another company they have been stolen.

METAPHOR This ability to talk about, and reason about, concepts in terms of *other distinct kinds of concepts* is called **metaphor** and is pervasive in all languages. As a generative model, it is responsible for a large proportion of the polysemy in the language, including many of the senses that are listed in dictionaries as well as the more novel ones that are not.

Let's now consider the following example from the WSJ.

(16.55) GM killed the Fiero because it had dedicated a full-scale factory to...

The use of *kill* in this example roughly means to *put an end to* some kind of ongoing effort, or activity. In this case, the ongoing activity of building, marketing, and selling a particular kind of car. The metaphor underlying this use views activities as living things, allowing the termination to be viewed as a killing. Note, however, that this sentence does not say any of this. In particular, the PATIENT of the killing is a definite reference *the Fiero*. For the metaphor to make sense, this phrase must refer not to a particular car, but rather to an entire sales and production effort at GM. At a very high level, this is a case where the result of an entire effort, or process, is being used to refer to the process itself. This is an example of **metonymy**, referring to a concept by mentioning a concept closely related to it. Like metaphor, metonymy is pervasive and goes mostly unnoticed in natural settings.

16.5 SUMMARY

This chapter has covered a wide range of issues concerning the meanings associated with lexical items. The following are among the highlights:

- Lexical semantics is the study of the systematic meaning-related connections among lexemes, and the internal meaning-related structure of individual lexemes.
- Homonymy refers to lexemes with the same form but unrelated meanings.
- Polysemy refers to the notion of a single lexeme with multiple related meanings.
- Synonymy holds between different lexemes with the same meaning.
- Hyponymy relations hold between lexemes that are in class-inclusion relationship.
- Semantic fields are used to capture semantic connections among groups of lexemes drawn from a single domain.
- WordNet is a large database of lexical relations for English words.
- Thematic roles abstract away from the specifics of deep semantic roles by generalizing over similar roles across classes of verbs.
- Semantic selection restrictions allow lexemes to post constraints on the semantic properties of the constituents that accompany them in sentences.
- Primitive decomposition allows permits the representation of the meanings of individual lexemes in terms of finite sets of sub-lexical primitives.
- Generative devices such as metaphor and metonymy are pervasive, and produce novel meanings that can not in principle be captured in a static lexicon.

BIBLIOGRAPHICAL AND HISTORICAL NOTES

Lyons (1977) and Cruse (1986) are classic linguistics texts on lexical semantics. Collections describing computational work on lexical semantics can be found in (Pustejovsky and Bergler, 1992; Saint-Dizier and Viegas, 1995; Klavans, 1995).

Martin (1986) and Copestake and Briscoe (1995) discuss computational approaches to the representation of polysemy. The most comprehensive collection of work concerning WordNet can be found in (Fellbaum, 1998). There have been many efforts to use existing dictionaries as lexical resources. One of the earliest was Amsler's (1980, 1981) use of the Merriam Webster dictionary. More recently, the machine readable version of Longman's Dictionary of Contemporary English has been used in a number of systems (Boguraev and Briscoe, 1989).

Thematic roles, or case roles, can be traced back to work by Fillmore (1968) and (Gruber, 1965b). Fillmore's work had an enormous and immediate impact on work in natural language processing. For a considerable period of time, nearly all work in natural language understanding used some version of Fillmore's case roles. Much of the early work in this vein was due to Simmons (1973b, 1978, 1983).

Work on selection restrictions as a way of characterizing semantic well-formedness began with (Katz and Fodor, 1963). McCawley (1968) was the first to point out that selection restrictions could not be restricted to a finite list of semantic features, but had to be drawn from a larger base of unrestricted world knowledge.

Lehrer (1974) is a classic text on semantic fields. More recent papers addressing this topic can be found in (Lehrer and Kittay, 1992). Baker *et al.* (1998) describe ongoing work on the FrameNet project.

The use of primitives, components, and features to define lexical items is ancient. Nida (1975) presents a comprehensive overview of work on componential analysis. Wierzbicka (Wierzbicka, 1996) has long been a major advocate of the use of primitives in linguistic semantics. Another prominent effort has been Jackendoff's Conceptual Semantics (Jackendoff, 1983a, 1990) work which combines thematic roles and primitive decomposition. On the computational side, Schank's Conceptual Dependency Schank (1972) remains the most widely used set of primitives in natural language processing. Wilks (1975a) was an early promoter of the use of primitives in machine translation, as well natural language understanding in general. More recently, Dorr (1993, 1992) has made considerable computational use of Jackendoff's framework in her work on machine translation.

An influential collection of papers on metaphor can be found in (Ortony, 1993). Lakoff and Johnson (1980) is the classic work on conceptual metaphor and metonymy. Pustejovsky (1995) introduced the notion of the *Generative Lexicon*, a conceptual framework that rejects the notion of the lexicon as a static repository in favor of a more dynamic view. Russell (1976)

presents one of the earliest computational approach to metaphor. Additional early work can be found in (DeJong and Waltz, 1983; Wilks, 1978; Hobbs, 1979b). More recent computational efforts to analyze metaphor can be found in (Fass, 1988, 1991; Martin, 1990; Veale and Keane, 1992; Iverson and Helmreich, 1992; Chandler, 1991). Martin (1996) presents a survey of computational approaches to metaphor and other types of figurative language.

EXERCISES

16.1 Collect three definitions of ordinary non-technical English words from a dictionary of your choice that you feel are flawed in some way. Explain the nature of the flaw and how it might be remedied.

16.2 Download and install the current version of WordNet.

16.3 Give a detailed account of similarities and differences among the following set of lexemes: *imitation*, *synthetic*, *artificial*, *fake* and *simulated*.

Examine the entries for these lexemes in WordNet (or some dictionary of your choice). How well does it reflect your analysis?

16.4 Consider the following examples from (McCawley, 1968).

My neighbor is a father of three.

?My buxom neighbor is a father of three.

What does the ill-formedness of the second example imply about how constituents satisfy, or violate, selection restrictions?

16.5 Find some articles about business, sports, or politics from your daily newspaper. Identify as many lexical metaphors and metonymies as you can in these articles. How many of these uses have reasonably close entries in either WordNet or your favorite dictionary?

16.6 [more to come]

17

WORD SENSE
DISAMBIGUATION AND
INFORMATION
RETRIEVAL

*Oh are you from Wales?
Do you know a fella named Jonah?
He used to live in whales for a while.*
Groucho Marx

This chapter introduces a number of topics related to **lexical semantic processing**. By this, we have in mind applications that make use of word meanings, but which are to varying degrees decoupled from the more complex tasks of compositional sentence analysis and discourse understanding.

LEXICAL
SEMANTIC
PROCESSING

The first topic we cover, **word sense disambiguation**, is of considerable theoretical and practical interest. As we noted in Chapter 16, the task of word sense disambiguation is to examine word tokens in context and specify which sense of each word is being used. As we will see in the next two sections, making this vague definition operational is a non-trivial — there is no clear consensus as to exactly what the task is, or how it should be evaluated. Nevertheless, there are robust algorithms that can achieve high levels of accuracy under certain reasonable assumptions.

WORD SENSE
DISAMBIGUA-
TION

The second topic we cover, **information retrieval**, is an extremely broad field, encompassing a wide-range of topics pertaining to the storage, analysis, and retrieval of all manner of media (Baeza-Yates and Ribeiro-Neto, 1999). Our concern in this chapter is solely with the storage and retrieval of text documents in response to users requests for information. We are interested in approaches in which users' needs are expressed as words, and documents are represented in terms of the words they contain. Section 17.3 presents the **vector space model**, a well-established approach used in most current systems, including most Web search engines.

INFORMATION
RETRIEVAL

17.1 SELECTION RESTRICTION-BASED DISAMBIGUATION

For the most part, our discussions of compositional semantic analyzers in Chapter 15 ignored the issue of lexical ambiguity. By now it should be clear that this is not a reasonable approach. Without some means of selecting correct senses for the words in the input, the enormous amount of homonymy and polysemy in the lexicon will quickly overwhelm any approach in an avalanche of competing interpretations. As with syntactic part-of-speech tagging, there are two fundamental approaches to handling this ambiguity problem. In the first approach, the selection of correct senses occurs during semantic analysis as a side-effect of the elimination of ill-formed representations composed from an incorrect combination of senses. In the second approach, sense disambiguation is performed as a stand-alone task independent of, and prior to, compositional semantic analysis. This section discusses the role of selection restrictions in the former approach. The stand-alone approach is discussed in detail in 17.2.

Selection restrictions and type hierarchies are the primary knowledge-sources used to perform disambiguation in most integrated approaches. In particular, they are used to rule out inappropriate senses and thereby reduce the amount of ambiguity present during semantic analysis. If we assume an integrated rule-to-rule approach to semantic analysis, then selection restrictions can be used to block the formation of component meaning representations that contain violations. By blocking such ill-formed components, the semantic analyzer will find itself dealing with fewer ambiguous meaning representations. This ability to focus on correct senses by eliminating flawed representations that result from incorrect senses can be viewed as a form of indirect word sense disambiguation. While the linguistic basis for this approach can be traced back to the work of Katz and Fodor (1963), the most sophisticated computational exploration of it is due to Hirst (1987).

As an example of this approach, consider the following pair of WSJ examples, focusing solely on their use of the lexeme *dish*.

- (17.1) “In our house, everybody has a career and none of them includes washing dishes”, he says.
- (17.2) In her tiny kitchen at home, Ms. Chen works efficiently, stir-frying several simple dishes, including braised pig’s ears and chicken livers with green peppers.

These examples make use of two polysemous senses of the lexeme *dish*. The first refers to the physical objects that we eat from, while the second refers to

the actual meals or recipes. The fact that we perceive no ambiguity in these examples can be attributed to the selection restrictions imposed by *wash* and *stir-fry* on their PATIENT roles, along with the semantic type information associated with the two senses of *dish*. More specifically, the restrictions imposed by *wash* conflict with the food sense of *dish* since it does not denote something that is normally washable. Similarly, the restrictions on *stir-fry* conflict with the artifact sense of *dish*, since it does not denote something edible. Therefore, in both of these cases *the predicate selects the correct sense* of an ambiguous argument by eliminating the sense that fails to match one of its selection restrictions.

Now consider the following WSJ and ATIS examples, focusing on the ambiguous predicate *serve*.

(17.3) Well, there was the time they served green-lipped mussels from New Zealand.

(17.4) Which airlines serve Denver?

(17.5) Which ones serve breakfast?

Here the sense of *serve* in 17.3 requires some kind of food as its PATIENT, the sense in 17.4 requires some kind of geographical or political entity, and the sense in the last example requires a meal designator. If we assume that *mussels*, *Denver* and *breakfast* are unambiguous, then in it is the arguments in these examples that select the appropriate sense of the verb.

Of course, there are also cases where both the predicate and the argument have multiple senses. Consider the following BERP example.

(17.6) I'm looking for a restaurant that serves vegetarian dishes.

Restricting ourselves to three senses of *serve* and two senses of *dish* yields six possible sense combinations in this example. However, since only one combination of the six is free from a selection restriction violation, determining the correct sense of both *serve* and *dish* is straightforward. In particular, the predicate and argument mutually select the correct senses.

Before moving on, we should note there will always be examples like the following where the available selection restrictions are too general to uniquely select a correct sense.

(17.7) What kind of dishes do you recommend?

In cases like this we either have to rely on the stand-alone methods discussed in 17.2, or knowledge of the broader discourse context, as will be discussed in Chapter 18.

Although there are a wide variety of ways to integrate this style of disambiguation into a semantic analyzer, the most straightforward approach follows the rule-to-rule strategy introduced in Chapter 15. In this integrated approach, fragments of meaning representations are composed and checked for selection restriction violations as soon as their corresponding syntactic constituents are created. Those representations that contain selection restriction violations are eliminated from further consideration.

This approach requires two additions to the knowledge structures used in our semantic analyzers: access to hierarchical type information about the arguments, and semantic selection restriction information about the arguments to predicates. Recall from Chapter 16, that both of these can be encoded using knowledge from WordNet. The first is available in form of the hypernym information about the heads of the meaning structures being used as arguments to predicates. Similarly, selection restriction information about argument roles can be encoded by associating the appropriate WordNet synsets with the arguments to each predicate-bearing lexical item. Exercise ?? asks you to explore this approach in more detail.

Limitations of Selection Restrictions

Not surprisingly, there are a number of practical and theoretical problems with this use of selection restrictions. The first symptom of these problems is the fact that there are many perfectly well-formed, interpretable, sentences that contain obvious violations of selection restrictions. Therefore, any approach based on a strict *elimination* of such interpretations is in serious trouble.

Consider the following WSJ example.

(17.8) But it fell apart in 1931, perhaps because people realized you can't eat gold for lunch if you're hungry.

The phrase *eat gold* clearly violates the selection restriction that *eat* places on its PATIENT role. Nevertheless, this example is perfectly well-formed. The key is the negative environment set up by *can't* prior to the violation of the restriction. This example makes it clear that any purely local, or rule-to-rule, analysis of selection restrictions will fail when a wider context makes the violation of a selection restriction acceptable, as in this case.

A second problem with selection restrictions is illustrated by the following example.

(17.9) In his two championship trials, Mr. Kulkarni ate glass on an empty stomach, accompanied only by water and tea.

Although the event described in this example is somewhat unusual, the sentence itself is not semantically ill-formed, despite the violation of *eat*'s selection restriction. Examples such as this illustrate the fact that thematic roles and selection restrictions are merely loose approximations of the deeper concepts they represent. They can not hope to account for uses such as this that require deeper commonsense knowledge about what eating is all about. At best, they reflect the idea that the things that are eaten are normally edible.

Finally, as discussed in Chapter 16, metaphoric and metonymic uses challenge this approach as well. Consider the following WSJ example.

(17.10) If you want to kill the Soviet Union, get it to try to eat Afghanistan.

Here the typical selection restrictions on the PATIENTS of both *kill* and *eat* will eliminate all possible literal senses leaving the system with no possible meanings. In many systems, such a situation serves to trigger alternative mechanisms for interpreting metaphor and metonymy (Fass, 1997).

As Hirst (1987) observes, examples like these often result in the elimination of all senses, bring semantic analysis to a halt. One approach to alleviating this problem is to adopt the view of selection restrictions as preferences, rather than rigid requirements. Although there have been many instantiations of this approach over the years (Wilks, 1975c, 1975b, 1978), the one that has received the most thorough empirical evaluation is Resnik's (1998) work, which uses the notion of a *selectional association* introduced on page ?? . Recall that this notion uses an empirically derived measure of the strength of association between a predicate and a class dominating the argument to the predicate.

A simplified version of Resnik's disambiguation algorithm is shown in Figure 17.1. The basic notion behind this algorithm is to select as the correct sense for the argument, the one that has the highest selectional association between one of its ancestor hypernyms and the predicate. Resnik (1998) reports an average of 44% correct with this technique for verb-object relationships, a result that is an improvement over a most frequent sense baseline. A limitation of this approach is that it only addresses the case where the predicate is unambiguous and *selects* the correct sense of the argument. A more complex decision criteria would be needed for the more likely situation where both the predicate and argument are ambiguous.


```

function SA-WSD(pred, arg) returns sense
  best-association ← Minimum possible selection association
  for each sense in senses of arg do
    for each hypernym in hypernyms of sense do
      new ← Selectional association between hyp and pred
      if new > best-association then
        best-association ← new
        best-sense ← sense
      end
    end
  end
  return best-sense

```

Figure 17.1 Resnik's (1998) selectional association-based word sense disambiguation algorithm. The selection association between all the hypernyms of all the senses of the target argument and the predicate are computed. The sense with the most closely associated hypernym is selected.

17.2 ROBUST WORD SENSE DISAMBIGUATION

The selection restriction approach to disambiguation has too many requirements to be useful in large-scale practical applications. Even with the use of WordNet, the requirements of complete selection restriction information for all predicate roles, and complete type information for the senses of all possible fillers are unlikely to be met. In addition, as we saw in Chapters 10, 12, and 15, the availability of a complete and accurate parse for all inputs is unlikely to be met in environments involving unrestricted text.

To address these concerns, a number of robust disambiguation systems with more modest requirements have been developed over the years. As with part-of-speech taggers, these systems are designed to operate in a stand-alone fashion and make minimal assumptions about what information will be available from other processes.

Machine Learning Approaches

In machine learning approaches, systems are *trained* to perform the task of word sense disambiguation. In these approaches, what is learned is a classifier that can be used to assign as yet unseen examples to one of a fixed number of senses. As we will see, these approaches vary as to the nature

of the training material, how much material is needed, the degree of human intervention, the kind of linguistic knowledge used, and the output produced. What they all share is an emphasis on acquiring the knowledge needed for the task from data, rather than from human analysts. The principal question to keep in mind as we explore these systems is whether the method scales; that is, would it be possible to apply the method to a substantial part of the entire vocabulary of a language?

The Inputs: Feature Vectors

Before discussing the algorithms, we should first characterize the kind of inputs they expect. In most of these approaches, the initial input consists of the word to be disambiguated, which we will refer to as the **target** word, along with a portion of the text in which it is embedded, which we will call its **context**. This initial input is then processed in the following ways:

- The input is normally part-of-speech tagged using one of the high accuracy methods described in Chapter 8.
- The original context may be replaced with larger or smaller segments surrounding the target word.
- Often some amount of stemming, or more sophisticated morphological processing, is performed.
- Less often, some form of partial parsing, or dependency parsing, is performed to ascertain thematic or grammatical roles and relations.

After this initial processing, the input is then boiled down to a fixed set of features that capture information relevant to the learning task. This task consists of two steps: selecting the relevant linguistic features, and encoding them in a form usable in a learning algorithm. Fortunately, a simple **feature vector** consisting of numeric or nominal values can easily encode the most frequently used linguistic information, and is appropriate for use in most learning algorithms

FEATURE
VECTOR

The linguistic features used in training WSD systems can be roughly divided into two classes: collocational features and co-occurrence features. In general, the term **collocation** refers to a quantifiable position-specific relationship between two lexical items. Collocational features encode information about the lexical inhabitants of *specific* positions located to the left and right of the target word. Typical items in this category include the word, the root form of the word, and the word's part-of-speech. This type of feature is effective at encoding local lexical and grammatical information that can often accurately isolate a given sense.

COLLOCATION

As an example of this type of feature-encoding, consider the situation where we need to disambiguate the lexeme *bass* in the following example.

(17.11) An electric guitar and **bass** player stand off to one side, not really part of the scene, just as a sort of nod to gringo expectations perhaps.

A feature-vector consisting of the two words to the right and left of the target word, along with their respective parts-of-speech, would yield the following vector.

[guitar, NN1, and, CJC, player, NN1, stand, VVB]

The second type of feature consists of co-occurrence data about neighboring words, ignoring their exact position. In this approach, the words themselves (or their roots) serve as features. The value of the feature is the number of times the word occurs in a region surrounding the target word. This region is most often defined as a fixed size window with the target word at the center. To make this approach manageable, a small number of frequently used content words are selected for use as features. This kind of feature is effective at capturing the general topic of the discourse in which the target word has occurred. This, in turn, tends to identify senses of a word that are specific to certain domains.

For example, a co-occurrence vector consisting of the 12 most frequent content words from a collection of *bass* sentences drawn from the WSJ corpus would have the words as features: *fishing, big, sound, player, fly, rod, pound, double, runs, playing, guitar, band*. Using these words as features with a window size of 10, Example 17.11 would be represented by the following vector.

[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0]

As we will see, most robust approaches to sense disambiguation make use of a combination of both collocational and co-occurrence features.

Supervised Learning Approaches

In supervised approaches, a sense disambiguation system is learned from a representative set of labeled instances drawn from the same distribution as the test set to be used. This is a straightforward application of the **supervised learning** approach to creating a classifier. In such approaches, a learning system is presented with a training set consisting of feature-encoded inputs *along with their appropriate label, or category*. The output of the system is a classifier system capable of assigning labels to new feature-encoded inputs.

METHODOLOGY BOX: EVALUATING WSD SYSTEMS

The basic metric used in evaluating sense disambiguation systems is simple precision: the percentage of words that are tagged correctly. The primary baseline against which this metric is compared is the **most frequent sense** metric: how well would a system do if it simply chose the most frequent sense of a word.

The use of precision requires access to the correct answers to the words in a test set. Fortunately, two large sense-tagged corpora are now available: the SEMCOR corpus (Landes *et al.*, 1998), which consists of a portion of the Brown corpus tagged with WordNet senses, and the SENSEVAL corpus (Kilgarriff and Rosenzweig, 2000), which is a tagged corpus derived from the HECTOR corpus and dictionary project.

A number of issues must be taken into account in comparing results across systems. The main issue concerns the nature of the senses used in the evaluation. Two approaches have been followed over the years: coarse distinctions among homographs, such as the musical and fish senses of *bass*, and fine-grained sense distinctions such as those found in traditional dictionaries. Unfortunately, there is no standard way of comparing results across these two kinds of efforts, or across efforts using different dictionaries.

Dictionary senses provide the opportunity for a more fine-grained scoring metric than simple precision. For example, confusing a particular musical sense of *bass* with a fish sense, is clearly worse than confusing it with another musical sense. This observation gives rise to a notion of **partial credit** in evaluating these systems. With such a metric, an exact sense-match would receive full credit, while selecting a broader sense would receive partial credit. Of course, this kind of scheme is entirely dependent on the organization of senses in the particular dictionary being used.

Standardized evaluation frameworks for word sense disambiguation systems are now available. In particular, the SENSEVAL effort (Kilgarriff and Palmer, 2000), provides the same kind of evaluation framework for sense disambiguation, that the MUC (Sundheim, 1995b) and TREC (Voorhees and Harman, 1998) evaluations have provided for information extraction and information retrieval.

Bayesian classifiers (Duda and Hart, 1973), decision lists (Rivest, 1987), decision trees (Quinlan, 1986), neural networks (Rumelhart *et al.*, 1986), logic learning systems (Mooney, 1995), and nearest neighbor methods (Cover and Hart, 1967) all fit into this paradigm. We will restrict our discussion to the naive Bayes and decision list approaches, since they have been the focus of considerable work in word sense disambiguation.

NAIVE BAYES

The **naive Bayes** classifier approach to WSD is based on the premise that choosing the best sense for an input vector amounts to choosing the most probable sense given that vector. In other words:

$$\hat{s} = \operatorname{argmax}_{s \in S} P(s|V) \quad (17.12)$$

In this formula, S denotes the set of senses appropriate for the target associated with this vector. As is almost always the case, it would be difficult to collect statistics for this equation directly. Instead, we rewrite it in the usual Bayesian manner as follows:

$$\hat{s} = \operatorname{argmax}_{s \in S} \frac{P(V|s)P(s)}{P(V)} \quad (17.13)$$

Of course, the data available that associates specific vectors with senses is too sparse to be useful. What is provided in abundance in the training set is information about individual feature-value pairs in the context of specific senses. Therefore, we can make the same independence assumption that has served us well in part-of-speech tagging, speech recognition, and probabilistic parsing — assume that the features are independent of one another. Making this assumption yields the following equation.

$$P(V|s) = \prod_{j=1}^n P(v_j|s) \quad (17.14)$$

Given this equation, *training* a Naive Bayes classifier amounts to collecting counts of the individual feature-value statistics with respect to each sense of the target word. The term $P(s)$ is the prior for each sense, which just corresponds to the proportion of each sense in the training set. Finally, since $P(V)$ is the same for all possible senses it does not effect the final ranking of senses, leaving us with the following.

$$\hat{s} = \operatorname{argmax}_{s \in S} P(s) \prod_{j=1}^n P(v_j|s) \quad (17.15)$$

Of course, all the issues discussed in Chapter 8 with respect to zero counts and smoothing apply here as well.

Rule		Sense
<i>fish</i> within window	⇒	bass ¹
<i>striped bass</i>	⇒	bass ¹
<i>guitar</i> within window	⇒	bass ²
<i>bass player</i>	⇒	bass ²
<i>piano</i> within window	⇒	bass ²
<i>tenor</i> within window	⇒	bass ²
<i>sea bass</i>	⇒	bass ¹
<i>play/V bass</i>	⇒	bass ²
<i>river</i> within window	⇒	bass ¹
<i>violin</i> within window	⇒	bass ²
<i>salmon</i> within window	⇒	bass ¹
<i>on bass</i>	⇒	bass ²
<i>bass are</i>	⇒	bass ¹

Figure 17.2 An abbreviated decision list for disambiguating the fish sense of bass from the music sense. (Adapted from (Yarowsky, 1996))

In a large experiment evaluating a number of supervised learning algorithms, Mooney (1996) reports that a naive-Bayes classifier and a neural network achieved the highest performance, both achieving around 73% correct in assigning one of 6 senses to a corpus of examples of the word *line*.

Decision list classifiers can be viewed as a simplified variant of decision trees. In a decision list classifier, a sequence of tests is applied to each vector encoded input. If a test succeeds, then the sense associated with that test is applied to the input and returned. If the test fails, then the next test in the sequence is applied. This continues until the end of the list, where a default test simply returns the majority sense. Figure 17.2 shows a portion of a decision list for the task of discriminating the fish sense of *bass* from the music sense.

Learning a decision list classifier consists of creating a good sequence of tests based on the characteristics of the training data. There are wide number of methods that can be used to create such lists. Yarowsky (1994) employs an extremely simple technique that yields excellent results in this domain. In this approach, all possible feature-value pairs are used to create tests. These individual tests are then ordered according to their individual accuracy on the training set, where the accuracy of a test is based on its

DECISION
LIST
CLASSIFIERS

log-likelihood ratio:

$$\text{Abs}\left(\text{Log}\left(\frac{P(\text{Sense}_1|f_i = v_j)}{P(\text{Sense}_2|f_i = v_j)}\right)\right) \quad (17.16)$$

The decision list is created from these tests by simply ordering the tests in the list according to this measure, with each test returning the appropriate sense. Yarowsky (1996) reports that this technique consistently achieves over 95% correct on a wide variety of binary decision tasks.

We should note that this training method differs quite a bit from the standard decision list learning algorithm. For the details and theoretical motivation for that approach see (Rivest, 1987; Russell and Norvig, 1995).

Bootstrapping Approaches

BOOTSTRAP-
PING
APPROACH

Not surprisingly, a major problem with supervised approaches is the need for a large sense-tagged training set. The **bootstrapping approach** (Hearst, 1991; Yarowsky, 1995) eliminates the need for a large training set by relying on a relatively small number of instances of each sense for each lexeme of interest. These labeled instances are used as *seeds* to train an initial classifier using any of the supervised learning methods mentioned in the last section. This initial classifier is then used to extract a larger training set from the remaining untagged corpus. Repeating this process results in a series of classifiers with improving accuracy and coverage.

The key to this approach lies in its ability to create a larger training set from a small set of seeds. To succeed, it must include only those instances in which the initial classifier has a high degree of confidence. This larger training set is then used to create a new more accurate classifier with broader coverage. With each iteration of this process, the training corpus grows and the untagged corpus shrinks. As with most iterative methods, this process can be repeated until some sufficiently low error-rate on the training set is reached, or until no further examples from the untagged corpus are above threshold.

The initial seed set used in these bootstrapping methods can be generated in a number of ways. Hearst (1991) generates a seed set by hand labeling a small set of examples from the initial corpus. This approach has three major advantages:

- There is a reasonable certainty that the seed instances are correct, thus ensuring that the learner does not get off on the wrong foot
- The analyst can make some effort to choose examples that are not only correct, but in some sense prototypical of each sense.

<p>Kluccevsek plays Giuliani or Titano piano accordions with the more flexible, more difficult free bass rather than the traditional Stradella bass with its preset chords designed mainly for accompaniment.</p> <p>We need more good teachers – right now, there are only a half a dozen who can play the free bass with ease.</p> <p>An electric guitar and bass player stand off to one side, not really part of the scene, just as a sort of nod to gringo expectations perhaps.</p> <p>When the New Jersey Jazz Society, in a fund-raiser for the American Jazz Hall of Fame, honors this historic night next Saturday, Harry Goodman, Mr. Goodman's brother and bass player at the original concert, will be in the audience with other family members.</p>
<p>The researchers said the worms spend part of their life cycle in such fish as Pacific salmon and striped bass and Pacific rockfish or snapper.</p> <p>Associates describe Mr. Whitacre as a quiet, disciplined and assertive manager whose favorite form of escape is bass fishing.</p> <p>And it all started when fishermen decided the striped bass in Lake Mead were too skinny.</p> <p>Though still a far cry from the lake's record 52-pound bass of a decade ago, "you could fillet these fish again, and that made people very, very happy," Mr. Paulson says.</p> <p>Saturday morning I arise at 8:30 and click on "America's best-known fisherman," giving advice on catching bass in cold weather from the seat of a bass boat in Louisiana.</p>
<p>Figure 17.3 Samples of <i>bass</i> sentences extracted from the WSJ using the simple correlates <i>play</i> and <i>fish</i>.</p>

- It is reasonably easy to carry out.

A remarkably effective alternative technique is to simply search for sentences containing single words that are strongly correlated with the target senses. Yarowsky (1995) calls this the One Sense per Collocation constraint and presents results that show that it yields remarkably good results. For example, Figure 17.3 shows a partial result of a such a search for the strings "fish" and "play" in a corpus of *bass* examples drawn from the WSJ.

Yarowsky (1995) suggests two methods to select effective correlates: deriving them from machine readable dictionary entries, and selecting seeds using collocations statistics such as those described in Chapter 6. Putting all of this to the test, Yarowsky (1995) reports an average performance of 96.5% on a coarse binary sense assignment of 12 words.

Unsupervised Methods: Discovering Word Senses

Unsupervised approaches to sense disambiguation eschew the use of sense tagged data of any kind during training. In these approaches, feature-vector representations of unlabeled instances are taken as input and are then grouped into clusters according to a similarity metric. These clusters can then be represented as the average of their constituent feature-vectors, and labeled by hand with known word senses. Unseen feature-encoded instances can be classified by assigning them the word sense from the cluster to which they are closest according to the similarity metric.

AGGLOMERATIVE
CLUSTERING

Fortunately, clustering is a well-studied problem with a wide number of standard algorithms that can be applied to inputs structured as vectors of numerical values (Duda and Hart, 1973). The most frequently used technique in language applications is known as **agglomerative clustering**. In this technique, each of the N training instances is initially assigned to its own cluster. New clusters are then formed in a bottom-up fashion by successively merging the two clusters that are most similar. This process continues until either a specified number of clusters is reached, or some global goodness measure among the clusters is achieved. In cases where the number of training instances makes this method too expensive, random sampling can be used on the original training set (Cutting *et al.*, 1992b) to achieve similar results.

Of course, the fact that these unsupervised methods do not make use of hand-labeled data poses a number of challenges for evaluating the goodness of any clustering result. The following problems are among the most important ones that have to be addressed in unsupervised approaches.

- The correct senses of the instances used in the training data may not be known.
- The clusters are almost certainly heterogeneous with respect to the senses of the training instances contained within them.
- The number of clusters is almost always different from the number of senses of the target word being disambiguated.

Schütze's experiments (Schütze, 1992, 1998) constitute the most extensive application of unsupervised clustering to word sense disambiguation to date. Although the actual technique is quite involved, unsupervised agglomerative clustering is at the core of the method. As with the supervised approaches, the bulk of this work is directed at coarse binary distinctions. In this work, the first two problems are addressed through the use of pseudo-words and a hand-labeling of a small subset of the instances in each cluster.

The heterogeneity issue is addressed by assigning the majority sense to each of the induced clusters. Given this approach, the last problem is not an issue; the various discovered clusters are simply labeled with their majority sense. The fact that there may be multiple clusters with the same sense is not directly an issue in disambiguation.

Schütze's results indicate that for coarse binary distinctions, unsupervised techniques can achieve results approaching those of supervised and bootstrap methods. In most instances approaching the 90% range. As with most of the supervised methods, this method was tested on a small sample of words (10 pseudowords, and 10 real words).

Dictionary-Based Approaches

A major drawback with all of the approaches described above is the problem of scale. All require a considerable amount of work to create a classifier for each ambiguous entry in the lexicon. For this reason, most of the experiments with these methods report results ranging from 2 to 12 lexical items (The work of Ng and Lee (1996) is a notable exception reporting results disambiguating 121 nouns and 70 verbs). Scaling up any of these approaches to deal with all the ambiguous words in a language would be a large undertaking. Instead, attempts to perform large-scale disambiguation have focused on the use of **machine readable dictionaries**, of the kind discussed in Chapter 16. In this style of approach, the dictionary provides both the means for constructing a sense tagger, and the target senses to be used.

The first implementation of this approach is due to Lesk (1986). In this approach, all the sense definitions of the word to be disambiguated are retrieved from the dictionary. These senses are then compared to the dictionary definitions of all the remaining words in the context. The sense with the highest overlap with these context words is chosen as the correct sense. Note that the various sense definitions of the context words are simply lumped together in this approach. Lesk reports accuracies of 50-70% on short samples of text selected from Austen's *Pride and Prejudice* and an AP newswire article.

The problem with this approach is that dictionary entries for the various senses of target words are relatively short, and may not provide sufficient material to create adequate classifiers.¹ More specifically, the words used in the context and their definitions must have direct overlap with the words

¹ Indeed, Lesk (Lesk, 1986) notes that the performance of his system seems to roughly correlate with the length of the dictionary entries.

contained in the appropriate sense definition in order to be useful. One way to remedy this problem is to expand the list of words used in the classifier to include words related to, but not contained in their individual sense definitions. This can be accomplished by including words whose definitions make use of the target word. For example, the word *deposit* does not occur in the definition of *bank* in the American Heritage Dictionary (Morris, 1985). However, *bank* does occur in the definition of *deposit*. Therefore, the classifier for *bank* can be expanded to include *deposit* as a relevant feature.

Of course, just knowing that *deposit* is related to *bank* does not help much since we don't know to which of *bank*'s senses it is related. Specifically, to make use of *deposit* as a feature we have to know which sense of *bank* was being used in its definition. Fortunately, many dictionaries and thesauri include tags known as subject codes in their entries that correspond roughly to broad conceptual categories. For example, the entry for *bank* in the *Longman's Dictionary of Contemporary English* (LDOCE) (Procter, 1978) includes the subject code EC (Economics) for the financial senses of *bank*. Given such subject codes, we can guess that expanded terms with the subject code EC will be related to this sense of bank rather than any of the others. Guthrie *et al.* (1991) report results ranging of 47% correct for fine-grained LDOCE distinctions to 72% for more coarse distinctions.

Note that none of these techniques actually exploit the dictionary entries *as definitions*. Rather, they can be viewed as variants of the supervised learning approach, where the content of the dictionary is used to provide the tagged training materials.

17.3 INFORMATION RETRIEVAL

The field of information retrieval is of interest to us here due to its widespread adoption of word-based indexing and retrieval methods. Most current information retrieval systems are based on an extreme interpretation of the principle of compositional semantics. In these systems, the meaning of documents resides solely in the words that are contained within them. To revisit the Mad Hatter's quote from the beginning of Chapter 16, in these systems *I see what I eat* and *I eat what I see* mean precisely the same thing. The ordering and constituency of the words that make up the sentences that make up documents play no role in determining their meaning. Because they ignore syntactic information, these approaches are often referred to as **bag of words** methods.

Before moving on, we need to introduce some new terminology. In information retrieval, a **document** refers generically to the unit of text indexed in the system and available for retrieval. Depending on the application, a document can refer to anything from intuitive notions like newspaper articles, or encyclopedia entries, to smaller units such as paragraphs and sentences. In Web-based applications, it can refer to a Web page, a part of a page, or to an entire Web-site. A **collection** refers to a set of documents being used to satisfy user requests. A **term** refers to a lexical item that occurs in a collection, but it may also include phrases. Finally, a **query** represents a user's information need expressed as a set of terms.

DOCUMENT

COLLECTION

TERM

QUERY

The specific information retrieval task that we will consider in detail is known as **ad hoc retrieval**. In this task, it is assumed that an unaided user poses a query to a retrieval system, which then returns a possibly ordered set of potentially useful documents. Several other related, lexically oriented, information retrieval tasks will be discussed in Section 17.4.

AD HOC
RETRIEVAL

The Vector Space Model

In the **vector space model** of information retrieval, documents and queries are represented as vectors of features representing the terms that occur within them (Salton, 1971). More properly, they are represented as vectors of features consisting of the terms that occur *within the collection*, with the value of each feature indicating the presence or absence of a given term in a given document. These vectors can be denoted as follows:

VECTOR
SPACE MODEL

$$\vec{d} = (t_1, t_2, t_3, \dots, t_N)$$

$$\vec{q} = (t_1, t_2, t_3, \dots, t_N)$$

In this notation, the various t features represent the N terms that occur in the collection. Let's first consider the case where these features take on the value of one or zero, indicating the presence or absence of a term in a document or query. Given this approach, a simple way to compare a document to a query, or another document, is to sum up the number of terms they have in common, as in the following equation.

$$s(\vec{q}_k, \vec{d}_j) = \sum_{i=1}^N t_{i,k} \times t_{i,j} \quad (17.17)$$

Of course, a problem with the use of binary values for features is that it fails to capture the fact that some terms are more important to the meaning of a document than others. A useful generalization is to replace the ones

and zeroes with numerical **weights** that indicate the importance of the various terms in particular documents and queries. We can thus generalize our vectors as follows:

$$\vec{d}_j = (w_{1,j}, w_{2,j}, w_{3,j}, \dots, w_{n,j})$$

$$\vec{q}_k = (w_{1,k}, w_{2,k}, w_{3,k}, \dots, w_{n,k})$$

TERM-BY-
DOCUMENT

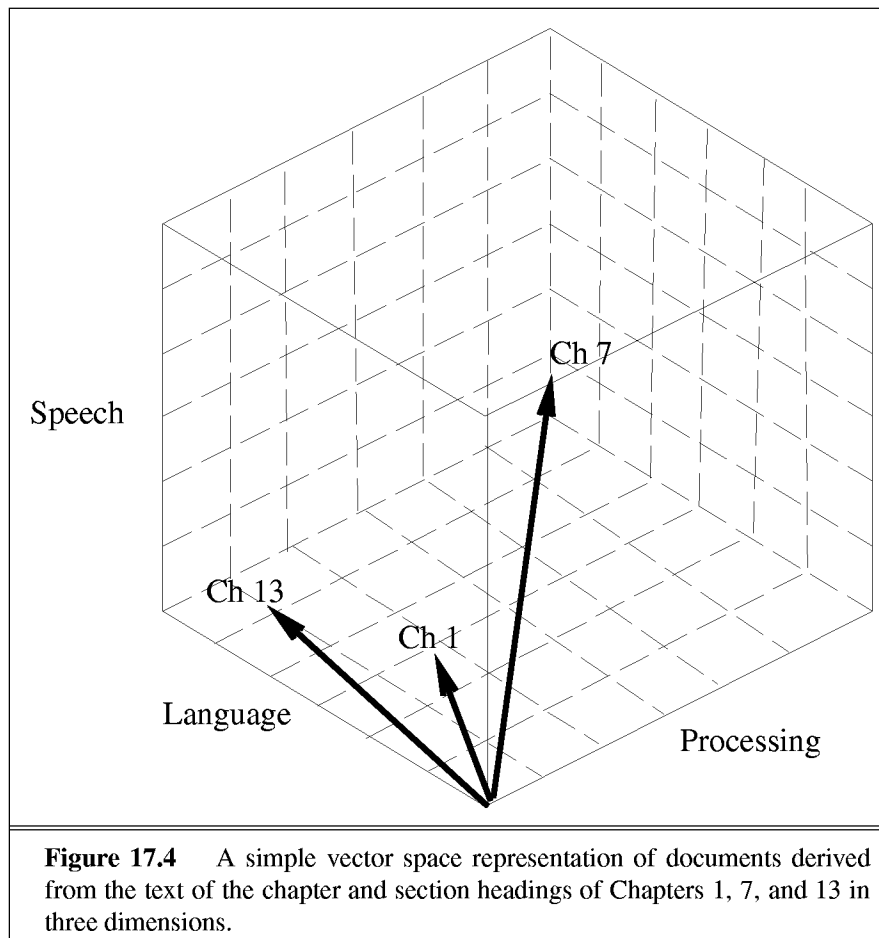
This characterization of individual documents as vectors of term weights allows us to view the document collection as a whole a matrix of weights, where $w_{i,j}$ represents the weight of term i in document j . This weight matrix is typically called a **term-by-document** matrix. Under this view, the columns of the matrix represent the documents in the collection, and the rows represent the terms.

A useful view of this model conceives of the features used to represent documents (and queries) as dimensions in a multi-dimensional space. Correspondingly, the weights that serve as values for those features serve to locate documents in that space. When a user's query is translated into a vector it denotes a point in that space. Documents that are located close to the query can then be judged as being more relevant than documents that are farther away.

This characterization of documents and queries as vectors, provides all the basic parts for an ad hoc retrieval system. A document retrieval system can simply accept a user's query, create a vector representation for it, compare it against the vectors representing all known documents, and sort the results. The result is a list of documents rank ordered by their similarity to the query.

Consider as an example of this approach, the space shown in Figure 17.4. This figure shows a simplified space consisting of the three dimensions corresponding to the terms *speech*, *language* and *processing*. The three vectors illustrated in this space represent documents derived from the chapter and section headings of Chapters 1, 7, and 13 of this text, which we'll denote as **Doc1**, **Doc7**, and **Doc13**, respectively. If we identify term weights with raw term frequency, then **Doc1** is represented by the vector (1,2,1), **Doc7** by (6,0,1), and **Doc13** by (0,5,1). As is clear from the figure, this space captures certain intuitions about how these chapters are related. Chapter 1, being general, is fairly similar to both Chapters 7 and 13. Chapters 7 and 13, on the other hand, are distant from one another since they cover a different set of topics.

Unfortunately, this particular instantiation of a vector space places too much emphasis on the absolute values of the various coordinates of each



document. For example, what is important about the *speech* dimension of the **Doc7**, is not the value 6 but rather that it is the dominant contributor to the meaning of that document. Similarly, the specific values of 1, 2, and 1 for **Doc1** are not important, what is important is that the three dimensions have roughly similar weights. It would be sensible, for example, to assume that a new document with weights 3, 6, and 3 would be quite similar to **Doc1** despite the magnitude differences in the term weights.

We can accomplish this effect by *normalizing* the document vectors. By normalizing, we simply mean converting all the vectors to a standard length. Converting to a unit length can be accomplished by dividing each of their dimensions by the overall length of the vector, which is defined as $\sum_{i=1}^N w_i^2$. This, in effect, eliminates the importance of the exact length of a

document's vector in the space, and emphasizes instead the direction of the document vector with respect to the origin.

Applying this technique to our three sample documents results in the following term-by-document matrix, A , where the columns represent **Doc1**, **Doc7** and **Doc13** and the rows represent the terms *speech*, *language*, and *processing*.

$$A = \begin{pmatrix} .41 & .81 & .41 \\ .98 & 0 & .16 \\ 0 & .98 & .19 \end{pmatrix}$$

You should verify that with this scheme, the normalized vectors for **Doc1** and our hypothetical (3,6,3) document end up as identical vectors.

Now let's return now to the topic of determining the similarity between vectors. Updating the similarity metric given earlier with numerical weights rather than binary values, gives us the following equation.

$$s(\vec{q}_k, \vec{d}_j) = \vec{q}_k \cdot \vec{d}_j = \sum_{i=1}^N w_{i,k} \times w_{i,j} \quad (17.18)$$

DOT
PRODUCT

This equation specifies what is known as the **dot product** between vectors. Now, in general, the dot product between two vectors is not particularly useful as a similarity metric, since it is too sensitive to the absolute magnitudes of the various dimensions. However, the dot product between vectors that have been normalized has a useful and intuitive interpretation: it computes the **cosine** of the angle between two vectors. When two documents are identical they will receive a cosine of one; when they are orthogonal (share no common terms) they will receive a cosine of zero.

COSINE

Note that if for some reason the vectors are not stored in a normalized form, then the normalization can be incorporated directly into the similarity measure as follows.

$$s(\vec{q}_k, \vec{d}_j) = \frac{\sum_{i=1}^N w_{i,k} \times w_{i,j}}{\sqrt{\sum_{i=1}^N w_{i,k}^2} \times \sqrt{\sum_{i=1}^N w_{i,j}^2}} \quad (17.19)$$

Of course, in situations where the document collection is relatively static and many queries are being performed, it makes sense to normalize the document vectors once and store them, rather than include the normalization in the similarity metric.

Let's consider how this similarity metric would work in the context of some small examples. Consider the carefully selected query consisting solely of the terms *speech*, *language* and *processing*. Converting this query to a vector and normalizing it results in the vector (.57, .57, .57). Computing

the cosines between this vector and our three document vectors shows that **Doc1** is closest with a cosine of .92, followed by **Doc13** with a cosine of .67, and finally **Doc7** with a cosine of .65. Not surprisingly, this ranking is in close accord with our intuitions about the relationship between this query and these documents.

Now consider a shorter query consisting solely of the terms *speech* and *processing*. Processing this query yields the normalized vector (.70, 0, .70). When the cosines are computed between this vector and our documents, **Doc7** is now the closest with a cosine of .80, followed by **Doc1** with a score of .58, with **Doc13** coming in a distant third with a cosine of .13.

Term Weighting

In practice, the method used to assign terms weights in the document and query vectors has an enormous impact on the effectiveness of a retrieval system. Two factors have proven to be critical in deriving effective term weights: term frequency within a single document, and the distribution of terms across a collection. We can begin with the simple notion that terms that occur frequently within a document may reflect its meaning more strongly than terms that occur less frequently and should thus have higher weights. In its simplest form, this factor is called **term frequency** and is simply the raw frequency of a term within a document (Luhn, 1957).

TERM
FREQUENCY

The second factor to consider is the distribution of terms across the collection as a whole. Terms that are limited to a few documents are useful for discriminating those documents from the rest of the collection. On the other hand, terms that occur frequently across the entire collection are less useful in discriminating among documents. What is needed therefore is a measure that favors terms that occur in fewer documents. The fraction N/n_i , where N is the total number of documents in the collection, and n is the number of documents in which term i occurs, provides exactly this measure. The fewer documents a term occurs in, the higher this weight. The lowest weight of 1 is assigned to terms that occur in all the documents. Due to the large number of documents in many collections, this measure is usually squashed with a log function leaving us with the following **inverse document frequency** term weight (Sparck Jones, 1972).

INVERSE
DOCUMENT
FREQUENCY

$$idf_i = \log\left(\frac{N}{n_i}\right) \quad (17.20)$$

Combining the term frequency factor with this factor results in a scheme

METHODOLOGY BOX: EVALUATING INFORMATION RETRIEVAL SYSTEMS

Information retrieval systems are evaluated with respect to the notion of **relevance** — *a judgment by a human* that a document is relevant to a query. A system's ability to retrieve relevant documents is assessed with a **recall** measure, as in Chapter 15.

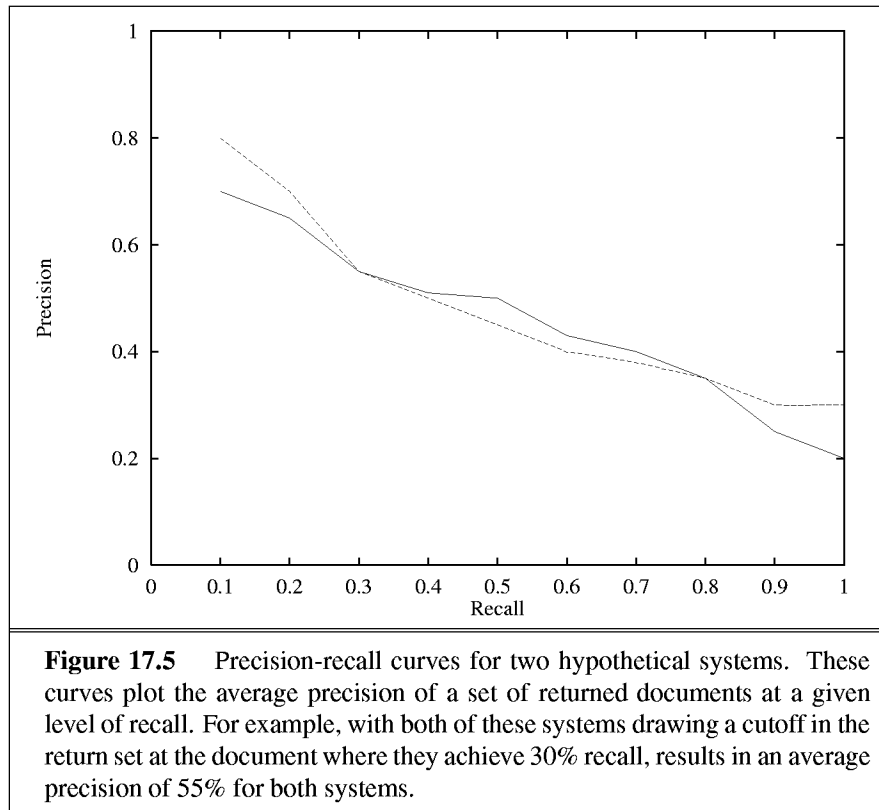
$$\text{Recall} = \frac{\text{\# of relevant documents returned}}{\text{total \# of relevant documents in the collection}}$$

Of course, a system can achieve 100% recall by simply returning all the documents in the collection. A system's accuracy is based on how many of the documents returned for a given query are actually relevant, which can be assessed by a **precision** metric.

$$\text{Precision} = \frac{\text{\# of relevant documents returned}}{\text{\# of documents returned}}$$

These measures are complicated by the fact that most systems do not make explicit relevance judgments, but rather rank their collection with respect to a query. To deal with this we can specify a set of cutoffs in the output, and measure average precision for the documents ranked above the cutoff. Alternatively, we can specify a set of recall levels and measure average precision at those levels. This latter method gives rise to what are known as precision-recall curves as shown in Figure 17.5. As these curves show, comparing the performance of two systems can be difficult. In this comparison, one system is better at both high and low levels of recall, while the other is better in the middle region. An alternative to these curves are metrics that attempt to combine recall and precision into a single value. The *F* measure introduced on page 576 is one such measure.

The U.S. government sponsored TREC (Text REtrieval Conference) evaluations have provided a rigorous testbed for the evaluation of a variety of information retrieval tasks and techniques. Like the MUC evaluations, TREC provides large document sets for both training and testing, along with a uniform scoring system. Training materials consist of sets of documents accompanied by sets of queries (called topics in TREC) and relevance judgments. Voorhees and Harman (1998) provides the details for the most recent meeting. Details of all of the meetings can be found at the TREC page on the National Institute of Standards and Technology Web site.



known as $tf \cdot idf$ weighting.

$$w_{i,j} = tf_{i,j} \times idf_i \quad (17.21)$$

That is, the weight of term i in the vector for document j is the product of its overall frequency in j with the log of its inverse document frequency in the collection. With some minor variations, this weighting scheme is used to assign term weights to documents in nearly all vector space retrieval models.

Despite the fact that we use the same representations for documents and queries, it is not at all clear that the same weighting scheme should be used for both. In many ad hoc retrieval settings such as Web search engines, user queries are not very much like documents at all. For example, an analysis of a very large set of queries (1,000,000,000 actually) from the AltaVista search engine reveals that the average query length is around 2.3 words (Silverstein *et al.*, 1998). In such an environment, the raw term frequency in the query is not likely to be a very useful factor. Instead, Salton and Buckley (1988) recommend the following formula for weighting query terms, where

$\text{Max}_j t f_{j,k}$ denotes the frequency of the most frequent term in document k .

$$w_{i,k} = \left(0.5 + \frac{0.5 t f_{i,k}}{\text{Max}_j t f_{j,k}} \right) \times idf_i \quad (17.22)$$

Term Selection and Creation

We have been assuming thus far that it is precisely the words that occur in a collection that will be used to index the documents in the collection. Two common variations on this assumption involve the use of **stemming**, and a **stop list**.

STEMMING

The notion of **stemming** takes us back to Chapter 3 and the topic morphological analysis. The basic question addressed by stemming is whether the morphological variants of a lexical item should be listed (and counted) separately, or whether they should be collapsed into a single root form. For example, without stemming, the terms *process*, *processing* and *processed* will be treated as distinct items with separate term frequencies in a term-by-document matrix; with stemming they will be conflated to the single term *process* with a single summed frequency count. The major advantage to using stemming is that it allows a particular query term to match documents containing any of the morphological variants of the term. The Porter stemmer (Porter, 1980) described Chapter 3 is the system most-used for this purpose retrieval from collections of English documents.

A significant problem with this approach is that it throws away useful distinctions. For example, consider the use of the Porter stemmer on documents and queries containing the words *stocks* and *stockings*. In this case, the Porter stemmer reduces these surface forms to the single term *stock*. Of course, the result of this is that queries concerning *stock prices* will return documents about *stockings*, and queries about *stockings* will find documents about *stocks*.² More technically, stemming may increase recall by finding documents with terms that are morphologically related to queries, but it may also reduce precision by returning semantically unrelated documents. For this reason, few Web search engines currently make use of stemming. Frakes and Baeza-Yates (1992) presents results from a series of experiments that explore the efficacy of stemming.

A second common technique is the use of stop lists, which address

² This example is motivated by some bad publicity received by a well-known search engine, when it returned some rather salacious sites containing extensive use of the term *stockings* in response to queries concerning *stock prices*. In response, a spokesman announced that their engineers were working hard on a solution to this strange problem with words.

the issue of what words should be allowed into the index. A **stop list** is a list of high frequency words that are eliminated from the representation of both documents and queries. Two motivations are normally given for this strategy: high frequency, closed-class, terms are seen as carrying little semantic weight and are thus unlikely to help with retrieval, and eliminating them can save considerable space in the inverted index files used to map from terms to the documents that contain them. The downside of using a stop list is that it makes it difficult to search for phrases that contain words in the stop list. For example, a common stop list derived from the Brown corpus presented in (Frakes and Baeza-Yates, 1992), would reduce the phrase *to be or not to be* to the phrase *not*.

STOP LIST

Homonymy, Polysemy and Synonymy

Since the vector space model is based solely on the use of simple terms, it is useful to consider the effect that various lexical semantic phenomena have on the model. Consider a query containing the word *canine* with its *tooth* and *dog* senses. A query containing *canine* will be judged similar to documents making use of either of these senses. However, given that users are probably only interested in one of these senses, the documents containing the other sense will be judged non-relevant. Homonymy and polysemy, therefore, have the effect of *reducing precision* by leading a system to return documents irrelevant to the users information need.

Now consider a query consisting of the lexeme *dog*. This query will be judged close to documents that make frequent use of the term *dog*, but may fail to match documents that use close synonyms like *canine*, as well as documents that use hyponyms such as *malamute*. Synonymy and hyponymy, therefore, have the effect of *reducing recall* by causing the retrieval system to miss relevant documents.

Note that it is inaccurate to state flatly that that polysemy reduces precision, and synonymy reduces recall since, as we discussed on page 648, both measures are relative to a fixed cutoff. As a result, every non-relevant document that rises above the cutoff due to polysemy takes up a slot in the fixed size return set, and may thus push a relevant document below threshold thus reducing recall. Similarly, when a document is missed due to synonymy, a slot is opened in the return set for a non-relevant document, potentially reducing precision as well.

Not surprisingly, these issues lead to the question of whether or not word sense disambiguation can help in information retrieval. The evidence

on this point is mixed, with some experiments reporting a sizable gain using disambiguation (Schütze and Pedersen, 1995), and others reporting either no gain, or a degradation in performance (Krovetz and Croft, 1992; Voorhees, 1998).

Improving User Queries

One of the most effective ways to improve retrieval performance is to find a way to improve user queries. The techniques presented in this section have been shown to varying degrees to be effective at this task.

RELEVANCE FEEDBACK

The single most effective way to improve retrieval performance in the vector space model is the use of **relevance feedback** (Rocchio, 1971). In this method, a user presents a query to the system and is presented with a small set of retrieved documents. The user is then asked to specify which of these documents appears relevant to their need. The user's original query is then reformulated based on the distribution of terms in the relevant and non-relevant documents that the user examined. This reformulated query is then passed to the system as a *new* query with the new results being shown to the user. Typically an enormous improvement is seen after a single iteration of this technique.

The formal basis for the implementation of this technique falls out directly from some of the basic geometric intuitions of the vector model. In particular, we would like to *push* the vector representing the user's original query toward the documents that have been found to be relevant, and away from the documents judged not relevant. This can be accomplished by adding an averaged vector representing the relevant documents to the original query, and subtracting an averaged vector representing the non-relevant queries.

More formally, let's assume that \vec{q}_i represents the user's original query, R is the number of relevant documents returned from the original query, and N is the number of non-relevant documents. In addition, assume that β and γ range from 0 to 1 and that $\beta + \gamma = 1$. Given these assumptions, the following represents a standard relevance feedback update formula.

$$\vec{q}_{i+1} = \vec{q}_i + \frac{\beta}{R} \sum_{j=1}^R \vec{d}_{ir} - \frac{\gamma}{N} \sum_{k=1}^N \vec{d}_{in}$$

The factors *beta* and γ in this formula represent parameters that can be adjusted experimentally. Intuitively, they represent how far the original vector should be pushed towards the relevant documents or away from the

non-relevant ones. Salton and Buckley (1990) report good results with $\beta = .75$ and $\gamma = .25$.

We should note that evaluating systems that use relevance feedback is rather tricky. In particular, an enormous improvement is often seen in the documents retrieved by the first reformulated query. This should not be too surprising since it includes the documents that the user has told the system were relevant. The preferred way to avoid this inflation is to only compute recall and precision measures for what is called the **residual collection**, the original collection without any of the documents shown to the user on any previous round. This usually has the effect of driving the system's raw performance below that achieved with the first query, since the most highly relevant documents have now been eliminated. Nevertheless, this is an effective technique to use when comparing distinct relevance feedback mechanisms.

RESIDUAL
COLLECTION

An alternative approach to query improvement focuses on the terms that comprise the query vector, rather than the query vector itself. In **query expansion**, the user's original query is expanded to include terms related to the original terms. This has typically been accomplished by adding terms chosen from lists of terms that are highly correlated with the user's original terms in the collection. Such highly correlated terms are listed in what is typically called a **thesaurus**, although since it is based on correlation, rather than synonymy, it is only loosely connected to the standard references that carry the same name.

QUERY
EXPANSION

THESAURUS

Unfortunately, it is usually the case that available thesaurus-like resources are not suitable for most collections. In **thesaurus generation**, a correlation-based thesaurus is generated automatically from all or a portion of the documents in the collection. Not surprisingly, one of the most popular methods used in thesaurus generation involves the use of **term clustering**. Recall, from our characterization of the term-by-document matrix that the columns in the matrix represent the documents and the rows represent the terms. Therefore, in thesaurus generation, the rows can be clustered to form sets of synonyms, which can then be added to the user's original query to improve its recall.

THESAURUS
GENERATION

TERM
CLUSTERING

This technique is typically instantiated in one of two ways: a thesaurus can be generated once from the document collection as a whole (Crouch and Yang, 1992), or sets of synonym-like terms can be generated dynamically from the returned set for the original query (Attar and Fraenkel, 1977). Note that this second approach entails far more effort, since in effect a small thesaurus is generated for the documents returned for every query, rather than once for entire collection.

17.4 OTHER INFORMATION RETRIEVAL TASKS

As noted earlier, ad-hoc retrieval is not the only word-based task in information retrieval. Some of the other more important ones include document categorization, document clustering, and text segmentation.

CATEGORIZA-
TION

The **categorization** task is to assign a new document to one of a pre-existing set of document classes. In this setting, the task of creating a classifier consists of discovering a useful characterization of the documents that belong in each class. Although this can be done by hand, the principal way to approach this problem is to use supervised machine learning. In particular, classifiers can be trained on a set of documents that have been labeled with the correct class. Not surprisingly, all the supervised learning methods introduced on page 634 for word sense disambiguation can be applied to this task as well.

FILTERING

ROUTING

When categorization is performed with the intent of then transmitting the document to a user or set of interested users it is usually referred to as **filtering** or **routing**. An interesting example of this is AT&T's 'How May I Help You' task where the goal is to classify a user's utterance into one of fifteen possible categories, such as third number billing, or collect call. Once the system has classified the call, the system routes the caller to an appropriate human operator. This task provides a good example of the need for *in vivo* evaluation mentioned earlier. The classification accuracy on this task approaches 80 %, despite the fact that the speech recognizer has a word accuracy rate of only around 50 % (Gorin *et al.*, 1997).

DOCUMENT
CLUSTERING

The categorization task assumes an existing classification, or clustering, of documents. By contrast, the task of **document clustering** is to create, or discover, a reasonable set of clusters for a given set of documents. As was the case word sense discovery, a reasonable cluster is defined as one that maximizes the within-cluster document similarity, and minimizes between-cluster similarity. There are two principal motivations for the use of this technique in an ad hoc retrieval setting: efficiency, and the **cluster hypothesis**.

The efficiency motivation arises from the enormous size of many modern document collections. Recall that the retrieval method described in the last section requires every query to be compared against every document in the collection. If a collection can be divided up into a set of N conceptually coherent clusters, then queries could first be compared against representations of each of the N clusters. Ordinary retrieval could then be applied only

within the top cluster or clusters, thus saving the cost of comparing the query to the documents in all of the other more distant clusters.

The **cluster hypothesis** (Jardine and van Rijsbergen, 1971) takes this argument a step further by asserting that retrieval from a clustered collection will not only be more efficient, but will in fact improve retrieval performance in terms of recall and precision. The basic notion behind this hypothesis is that by separating documents according to topic, relevant documents will be found together in the same cluster, and non-relevant documents will be avoided since they will be reside in clusters that are not used for retrieval. Despite the plausibility of this hypothesis, there is only mixed experimental support for it. Results vary considerably based on the clustering algorithm and document collection in use (Willett, 1988; Shaw *et al.*, 1996).

CLUSTER
HYPOTHESIS

Finally, in **text segmentation**, larger documents are automatically broken down into smaller semantically coherent chunks. This is useful in domains where there are a significant number of large documents that cover a wide variety of topics. Text segmentation can be used to either perform retrieval below the document level, or to visually guide the user to relevant parts of retrieved documents. Again, not surprisingly, segmentation algorithms often make use of vector-like representations for the subparts of a larger document. Adjacent subparts that have similar cosines are more likely to about the same topic than adjacent segments with more distant cosines. Roughly speaking, such discontinuities in the similarity between adjacent text segments can be used to divide larger documents into subparts (Salton *et al.*, 1993; Hearst, 1997).

TEXT SEG-
MENTATION

17.5 SUMMARY

This chapter has explored two major areas of lexical semantic processing: word sense disambiguation and information retrieval.

- Word sense disambiguation systems assign word tokens in context to one of a pre-specified set of senses.
- Selection restriction-based approaches can be used to disambiguate both predicates and arguments.
- Selection restriction-based methods require considerable information about semantic roles restrictions and hierarchical type information about role fillers.

- Machine learning approaches to sense disambiguation make it possible to automatically create robust sense disambiguation systems.
- Supervised approaches use collections of texts annotated with their correct senses to train classifiers.
- Bootstrapping approaches permit the use of supervised methods with far fewer resources.
- Unsupervised, clustering-based, approaches attempt to discover representations of word senses from unannotated texts.
- Machine readable dictionaries facilitate the creation of broad-coverage sense disambiguators.
- The dominant models of information retrieval represent the meanings of documents and queries as bags of words.
- The vector space model views documents and queries as vectors in a large multidimensional space.
- The similarity between documents and queries, or other documents, can be measured by the cosine of the angle between the vectors.
- The values of the features of vectors is based on a combination of the frequency of terms within a document and the distribution of terms across the document.
- Polysemy and synonymy wreak havoc with word-based information retrieval systems, reducing both precision and recall.
- User queries can be improved through query reformulation using either relevance feedback or thesaurus-based query expansion.

BIBLIOGRAPHICAL AND HISTORICAL NOTES

Word sense disambiguation traces its roots to some of the earliest applications of digital computers. The notion of disambiguating a word by looking at small window around it was apparently first suggested by Warren Weaver (1955b), in the context of machine translation. Among the notions first proposed in this early period were the use of a thesaurus for disambiguation (Masterman, 1957), supervised training of Bayesian models for disambiguation (Madhu and Lytel, 1965), and the use of clustering in word sense analysis (Sparck Jones, 1986).

An enormous amount of work on disambiguation has been conducted within the context of AI-oriented natural language processing systems. It is

fair to say that most natural language analysis systems of this type exhibit some form of lexical disambiguation capability. However, a number of these efforts made word sense disambiguation a larger focus of their work. Among the most influential efforts were the efforts of Quillian (1968) and Simmons (1973b) with semantic networks, the work of Wilks with *Preference Semantics* (Wilks, 1975c, 1975b, 1975a)ks75, and the work of Small and Rieger (1982) and Riesbeck (1975) on word-based understanding systems. Hirst's ABSITY system (Hirst and Charniak, 1982; Hirst, 1986, 1988), which used a technique based on semantic networks called marker passing, represents the most advanced system of this type. As with these largely symbolic approaches, most connectionist approaches to word sense disambiguation have relied on small lexicons with hand-coded representations (Cottrell, 1985; Kawamoto, 1988).

We should note that considerable work on sense disambiguation has been conducted in the areas of Cognitive Science and psycholinguistics. Appropriately enough, it is generally described using a different name: lexical ambiguity resolution. Small *et al.* (1988) present a variety of papers from this perspective.

The earliest implementation of a robust empirical approach to sense disambiguation is due to Kelly and Stone (1975) who directed a team of that hand-crafted a set of disambiguation rules for 1790 ambiguous English words. Lesk (1986) was the first to use a machine readable dictionary for word sense disambiguation. The efforts at New Mexico State University using LDOCE are among the most extensive explorations of the use of machine readable dictionaries. Much of this work is described in (Wilks *et al.*, 1996). The problem of dictionary senses being too fine-grained or lacking an appropriate organization has been addressed in the work of (Dolan, 1994) and (Chen and Chang, 1998).

Modern interest in supervised machine learning approaches to disambiguation began with Black (1988), who applied decision tree learning to the task. The need for large amounts of annotated text in these methods led to investigations into the use of bootstrapping methods (Hearst, 1991; Yarowsky, 1995). The problem of how to weight and combine the disparate sources of evidence used in many robust systems is explored in (Ng and Lee, 1996) and (McRoy, 1992). There has been considerably less work in the area of unsupervised methods. The earliest attempt attempt to use clustering in the study of word senses is due to (Sparck Jones, 1986). Zernik (1991) successfully applied a standard information retrieval clustering algorithm to the problem, and provided an evaluation based on improvements in retrieval performance.

More extensive recent work on clustering can be found in (Pedersen and Bruce, 1997; Schütze, 1997, 1998).

Note that of all of these robust efforts, only three have attempted to exploit the power of mutually disambiguating all the words in a sentence. The system described in (Kelly and Stone, 1975) makes multiple passes over a sentence to take later advantage of easily disambiguated words; Cowie *et al.* (1992) use a simulated annealing model to perform a parallel search for a desirable set of senses; Veronis and Ide (1990) use inhibition and excitation in a neural network automatically constructed from a machine readable dictionary.

Ide and Veronis (1998) provide a comprehensive review of the history and current state of word sense disambiguation. (Ng and Zelle, 1997) provide a more focused review from a machine learning perspective. Wilks *et al.* (1996) describe a wide array of dictionary and corpus-based experiments, along with detailed descriptions of some very early work.

Luhn (1957) is generally credited with first advancing the notion of fully automatic indexing of documents based on their contents. Over the years Salton's SMART project (Salton, 1971) at Cornell developed or evaluated many of the most important notions in information retrieval including the vector model, term weighting schemes, relevance feedback, and the use of cosine as a similarity metric. The notion of using inverse document frequency in term weighting is due to (Sparck Jones, 1972). The original notion of relevance feedback is due to (Rocchio, 1971). An alternative to the vector model that we have not covered is the **probabilistic model**. Originally shown effective by Robinson and Sparck Jones (1976), a Bayesian network version of the probabilistic model is the basis for the widely used INQUERY system (Callan *et al.*, 1992).

PROBABILIS-
TIC
MODEL

The cluster hypothesis was introduced in (Jardine and van Rijsbergen, 1971). Willett (1988) provides a critical review of the major efforts in this area. Mather (1998) presents an algorithm-independent clustering metric that can be used to evaluate the performance of various clustering algorithms. A collection of papers on document categorization and its close siblings, filtering and routing, can be found in (Lewis and Hayes, 1994). Text segmentation has generally been investigated from one of two perspectives: approaches based on strong theories of discourse structure, and approaches based on lexical text cohesion (Morris and Hirst, 1991). Hearst (1997) describes a robust technique based on a vector model of lexical cohesion. Techniques based on strong discourse-models are discussed in Chapter 18 and Chapter 20.

An important extension of the vector space model known as **Latent Semantic Indexing** (LSI) (Deerwester *et al.*, 1990) uses the singular value decomposition method as means of *reducing the dimensionality* of vector models with the intent of discovering higher-order regularities in the original term-by-document matrix. Although LSI began life as a retrieval method, it has been applied to a wide variety of applications including models of lexical acquisition (Landauer and Dumais, 1997), question answering (Jones, 1997), and most recently, student essay grading (Landauer *et al.*, 1997).

Baeza-Yates and Ribeiro-Neto (1999) is a comprehensive text covering many of newest advances and trends in information retrieval. Frakes and Baeza-Yates (1992) is a more nuts and bolts text which includes a considerable amount of useful C code. Older classic texts include (Salton and McGill, 1983) and (van Rijsbergen, 1975). (Sparck Jones and Willett, 1997) includes many of the classic papers in the field. Current work is often published in the annual proceedings of the ACM Special Interest Group on Information Retrieval (SIGIR). The periodic TREC conference proceedings contain results from standardized evaluations organized by the U.S. government. The primary journals in the field are the *Journal of the American Society of Information Sciences*, *ACM Transactions on Information Systems*, *Information Processing and Management*, and *Information Retrieval*.

EXERCISES

Part IV

PRAGMATICS

Pragmatics is the study of (some parts of) the relation between language and context-of-use. Context-of-use includes such things as the identities of people and objects, and so pragmatics includes studies of how language is used to refer (and re-refer) to people and things. Context-of-use includes the discourse context, and so pragmatics includes studies of how discourses are structured, and how the listener manages to interpret a conversational partner in a conversation. This section explores algorithms for reference resolution, computational models for recovering the structure of monologue and conversational discourse, and models of how utterances in dialog are interpreted. This section also discusses the role of each of these models in building a conversational agent, as well as the design of the dialog manager component of such an agent. Finally, the section introduces natural language generation, focusing especially on the function of discourse.