Chapter 18
Computational Semantics

"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*

Semantic analysis

This chapter presents a principled computational approach to the problem of **semantic analysis** the process whereby meaning representations of the kind discussed in the last chapter are composed for linguistic expressions. The automated creation of accurate and expressive meaning representations necessarily involves a wide range of knowledge-sources and inference techniques. Among the sources of knowledge that are typically involved are the meanings of words, the conventional meanings associated with grammatical constructions, knowledge about the structure of the discourse, common-sense knowledge about the topic at hand and knowledge about the state of affairs in which the discourse is occurring.

Syntax-driven semantic analysis

The focus of this chapter is a kind of **syntax-driven semantic analysis** that is fairly modest in its scope. In this approach, meaning representations are assigned to sentences based solely on knowledge gleaned from the lexicon and the grammar. When we refer to an expression's meaning, or meaning representation, we have in mind a representation that is both context independent and free of inference. Representations of this type correspond to the traditional notion of literal meaning discussed in the previous chapter.

There are two motivations for proceeding along these lines: there are application domains, including question answering, where such primitive representations are sufficient to produce useful results, and these impoverished representations can serve as useful inputs to subsequent processes that can produce richer, more complete, meaning representations. Chs. 21 and 24 will discuss how these meaning representations can be used in processing extended discourses and dialogs.

18.1 Syntax-Driven Semantic Analysis

Principle of compositionality

The approach detailed in this section is based on the **principle of compositionality**. The key idea behind this approach is that the meaning of a sentence can be constructed from the meanings of its parts. When interpreted superficially this principle is somewhat less than useful. We know that sentences are composed of words, and that words

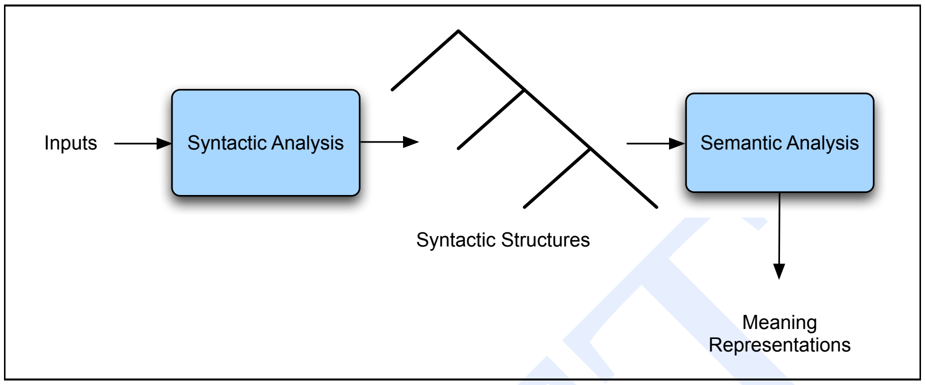

Figure 18.1 A simple pipeline approach to semantic analysis.

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, but also on the ordering and grouping of words, and on the relations among the words in the sentence. This is just another way 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 Ch. 12.

Let's begin by assuming that the syntactic analysis of an input sentence serves as the input to a semantic analyzer. Figure 18.1 illustrates an 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 flat chunks, feature structures, or dependency structures can also be used. For the remainder of this chapter we'll assume tree-like inputs.

Semantic analyzer

Before moving on, we should touch on the role of ambiguity in this story. As we've seen, ambiguous representations can arise from numerous sources including competing syntactic analyses, ambiguous lexical items, competing anaphoric references and as we'll see later in this chapter ambiguous quantifier scopes. In the syntax-driven approach presented here, we assume that syntactic, lexical and anaphoric ambiguities are not a problem. That is, we'll assume that some larger system is capable of iterating through the possible ambiguous interpretations and passing them individually to the kind of semantic analyzer described here.

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

(18.1) Franco likes Frasca.

Fig. 18.2 shows a simplified parse tree (lacking any feature attachments), along with a plausible 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

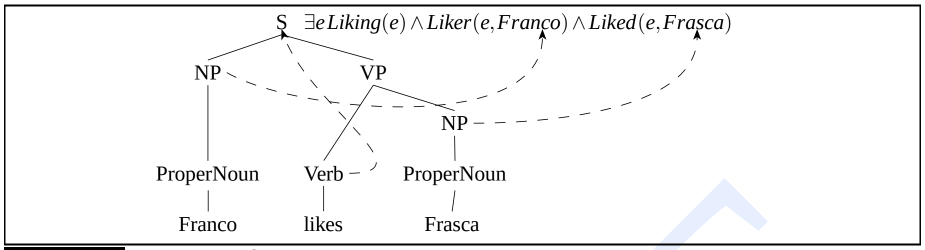

Figure 18.2 Parse tree for the sentence *Franco likes Frasca.*

skeletal meaning representation from the subtree corresponding to the verb *likes*. The analyzer would then retrieve or compose meaning representations corresponding to the two noun phrases in the sentence. Then using the representation acquired from the verb as a kind of template, the noun phrase meaning representations would be used to bind the appropriate variables in the verb representation, thus producing the meaning representation for the sentence as a whole.

Unfortunately, there are a number of serious difficulties with this simplified story. As described, the function used to interpret the tree in Fig. 18.2 must know, among other things, that it is the verb that carries the template upon which the final representation is based, 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 desired 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).

Rule-to-rule hypothesis

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 following section is an attempt to answer the following two questions:

- What does it mean for a syntactic constituent to have a meaning?
- What characteristics do these meanings have to have so that they can be composed into larger meanings?

18.2 Semantic Augmentations to CFG Rules

Semantic attachments

In keeping with the approach used in Ch. 16, we will begin by augmenting our context-free grammar rules with **semantic attachments**. These attachments are 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:

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

The semantic attachment to the basic context-free rule is shown in the $\{\ldots\}$ 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.

There are myriad ways to instantiate this style of rule-to-rule approach. Our semantic attachments could, for example, take the form of arbitrary programming language fragments. A meaning representation for a given derivation could then be constructed by passing the appropriate fragments to an interpreter in a bottom-up fashion and then storing the resulting representations as the value for the associated non-terminals.[1] Such an approach would allow us to create any meaning representation we might like. Unfortunately, the unrestricted power of this approach would also allow us to create representations that have no correspondence at all with the kind of formal logical expressions described in the last chapter. Moreover, this approach would provide us with very little guidance as to how to go about designing the semantic attachments to our grammar rules.

For these reasons, more principled approaches are typically used to instantiate the rule-to-rule approach. We'll introduce two such constrained approaches in this chapter. The first makes direct use of FOL and the λ-calculus notation introduced in Ch. 17. This approach essentially uses a logical notation to guide the creation of logical forms in a principled fashion. The second approach, described later in Sec. 18.4 is based on the feature-structure and unification formalisms introduced in Ch. 16.

To get started, let's take a look at a very basic example along with a simplified target semantic representation.

(18.2) Maharani closed.

$Closed(Maharani)$

Let's work our way bottom-up through the rules involved in this example's derivation. Starting with the proper noun, the simplest possible approach is to assign a unique FOL constant to it, as in the following.

$$ProperNoun \rightarrow Maharani \qquad \{Maharani\}$$

The non-branching *NP* rule that dominates this one doesn't add anything semantically, so we'll just copy the semantics of the *ProperNoun* up unchanged to the NP.

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

Moving on to the *VP*, the semantic attachment for the verb needs to provide the name of the predicate, specify its arity and provide the means to incorporate an argument once it's discovered. We'll make use of a λ-expression to accomplish these tasks.

1 Those familiar with compiler tools such as YACC and Bison will recognize this approach.

$$VP \rightarrow Verb \qquad \{Verb.sem\}$$

$$Verb \rightarrow closed \qquad \{\lambda x.Closed(x)\}$$

This attachment stipulates that the verb *closed* has a unary predicate *Closed* as its representation. The λ-notation gives us the means to leave unspecified, as the x variable, the entity that is closing. As with our earlier *NP* rule, the intransitive *VP* rule that dominates the verb simply copies upward the semantics of the verb below it.

Proceeding upward, it remains for the semantic attachment for the *S* rule to bring things together by inserting the semantic representation of the subject *NP* as the first argument to the predicate.

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

Since the value of $VP.sem$ is a λ-expression and the value of $NP.sem$ is a simply a FOL constant, we can create our desired final meaning representation by using λ-reduction to apply the $VP.sem$ to the $NP.sem$.

$$\lambda x.Closed(x)(Maharani)$$

$$Closed(Maharani)$$

This example illustrates a general pattern which will repeat itself throughout this chapter. The semantic attachments to our grammar rules will consist primarily of λ-reductions, where one element of an attachment serves as a functor and the rest serve as arguments to it. As we'll see, the real work resides in the lexicon where the bulk of the meaning representations are introduced.

Although this example illustrates the basic approach, the full story is a bit more complex. Let's begin by replacing our earlier target representation with one that is more in keeping with the neo-Davidsonian representations introduced in the last chapter, and by considering an example with a more complex noun phrase as its subject.

(18.3) Every restaurant closed.

The target representation for this example should be the following.

$$\forall x Restaurant(x) \Rightarrow (\exists e Closed(e) \wedge ClosedThing(e,x)$$

Clearly, the semantic contribution of the subject noun phrase in this example is much more extensive than in our previous one. In our earlier example, the FOL constant representing the subject was simply plugged into the correct place in *Closed* predicate via a single λ-reduction. Here the final result involves a complex intertwining of the content provided by the *NP* and the content provided by the *VP*. We'll have to do some work if we want rely on λ-reduction to produce what we want here.

The first step is to determine exactly what we'd like the meaning representation of *Every restaurant* to be. Let's start by assuming that *Every* invokes the $\forall$ quantifier and that *restaurant* specifies the category of concept that we're quantifying over, which we'll call the **restriction** of the noun phrase. Putting these together we might expect the meaning representation to be something like $\forall x Restaurant(x)$. Although this is a valid FOL formula its not a terribly useful one, since it says that everything is a restaurant. What's missing from it is the notion that noun phrases like *every restaurant* are normally embedded in expressions that stipulate something about the universally quantified variable. That is, we're probably trying to *say something* about all restaurants. This notion is traditionally referred to as the *NP*'s **nuclear scope**. In this case, the nuclear scope of this noun phrase is *closed*.

Restriction

Nuclear scope

We can capture these notions in our target representation by adding a dummy predicate, Q, representing the scope and attaching that predicate to the restriction predicate with an $\Rightarrow$ logical connective, leaving us with the following expression:

$$\forall x Restaurant(x) \Rightarrow Q(x)$$

Ultimately, what we need to do to make this expression meaningful is to replace Q with the logical expression corresponding to the nuclear scope. Fortunately, the λ-calculus can come to our rescue again. All we need to do is to permit λ-variables to range over FOL predicates as well as terms. The following expression captures exactly what we need.

$$\lambda Q.\forall x Restaurant(x) \Rightarrow Q(x)$$

The following series of grammar rules with their semantic attachments serve to produce this desired meaning representation for this kind of *NP*.

$$NP \rightarrow Det\ Nominal \quad \{Det.Sem(Nominal.Sem)\}$$

$$Det \rightarrow every \quad \{\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)\}$$

$$Nominal \rightarrow Noun \quad \{Noun.sem\}$$

$$Noun \rightarrow restaurant \quad \{\lambda x.Restaurant(x)\}$$

The critical step in this sequence involves the λ-reduction in the *NP* rule. This rule applies the λ-expression attached to the *Det* to the semantic attachment of the *Nominal*, which is itself a λ-expression. The following are the intermediate steps in this process.

$$\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)(\lambda x.Restaurant(x))$$

$$\lambda Q.\forall x \lambda x.Restaurant(x)(x) \Rightarrow Q(x)$$

$$\lambda Q.\forall x\ Restaurant(x) \Rightarrow Q(x)$$

The first expression is the expansion of the $Det.Sem(Nominal.Sem)$ semantic attachment to the *NP* rule. The second formula is the result of this λ-reduction. Note that this second formula has a λ-application embedded in it. Reducing this expression in place gives us the final form.

Having revised our semantic attachment for the subject noun phrase portion of our example, let's move to the *S* and *VP* and *Verb* rules to see how they need to change to accommodate these revisions. Let's start with the *S* rule and work our way down. Since the meaning of the subject *NP* is now a λ-expression, it makes sense to consider it as a functor to be called with the meaning of the *VP* as its argument. The following attachment accomplishes this.

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

Note that we've flipped the role of functor and argument from our original proposal for this *S* rule.

The last attachment to revisit is the one for the verb *close*. We need to update it to provide a proper event-oriented representation and to make sure that it is interfaces well with the new *S* and *NP* rules. The following attachment accomplishes both goals.

$$Verb \rightarrow close \qquad \{\lambda x.\exists e Closed(e) \wedge ClosedThing(e,x)\}$$

This attachment is passed unchanged to the *VP* constituent via the intransitive *VP* rule. It is then combined with the meaning representation of *Every restaurant* as dictated by the semantic attachment for the *S* given earlier. The following expressions illustrate the intermediate steps in this process.

$$\lambda Q.\forall x Restaurant(x) \Rightarrow Q(x)(\lambda y.\exists e Closed(e) \wedge ClosedThing(e,y))$$

$$\forall x Restaurant(x) \Rightarrow \lambda y.\exists e Closed(e) \wedge ClosedThing(e,y)(x)$$

$$\forall x Restaurant(x) \Rightarrow \exists e Closed(e) \wedge ClosedThing(e,x)$$

These steps achieve our goal of getting the *VP*'s meaning representation spliced in as the nuclear scope in the *NP*'s representation.

As is always the case with any kind of grammar engineering effort we now need to make sure that our earlier simpler examples still work. One area that we need to revisit is our representation of proper nouns. Let's consider them in the context of our earlier example.

(18.4) Maharani closed.

The *S* rule now expects the subject *NP*'s semantic attachment to be a functor applied to the semantics of the *VP*, therefore our earlier representation of proper nouns as FOL constants won't do. Fortunately, we can once again exploit the flexibility of the λ-calculus to accomplish what we need with the following expression.

$$\lambda x.x(Maharani)$$

This trick turns a simple FOL constant into a λ-expression, which when reduced serves to inject the constant into a larger expression. You should work through our original example with all of the new semantic rules to make sure that you can come up with the following intended representation:

$$\exists e Closed(e) \wedge ClosedThing(Maharani)$$

As one final exercise, let's see how this approach extends to an expression involving a transitive verb phrase, as in the following.

(18.5) Matthew opened a restaurant.

If we've done things correctly we ought to be able to specify the semantic attachments for transitive verb phrases, for the verb *open* and for the determiner *a*, while leaving the rest of our rules alone.

Let's start by modeling the semantics for the determiner *a* on our earlier attachment for *every*.

$$Det \rightarrow a \qquad \{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)\}$$

This rule differs from the attachment for *every* in two ways. First we're using the existential quantifier $\exists$ to capture the semantics of *a*. And second we've replaced the $\Rightarrow$ operator with a logical $\wedge$. The overall framework remains the same with the λ-variables *P* and *Q* standing in for the restriction and nuclear scopes to be filled in later. With this addition our existing *NP* rule will create the appropriate representation for *a restaurant*:

$$\lambda Q.\exists x Restaurant(x) \wedge Q(x)$$

Next let's move on to the *Verb* and *VP* rules. There are two arguments that need to be incorporated into the underlying meaning representation. One argument is available at the level of the transitive *VP* rule, and the second at the *S* rule. Let's assume the following form for the *VP* semantic attachment.

$$VP \rightarrow Verb\ NP \qquad \{Verb.Sem(NP.Sem)\}$$

This attachment assumes that the verb's semantic attachment will be applied as a functor to the semantics of its noun phrase argument. And let's assume for now that the representations we developed earlier for quantified noun phrases and proper nouns will remain unchanged. With these assumptions in mind, the following attachment for the verb *opened* will do what we want.

$$\begin{array}{l} Verb \rightarrow opened \\ \qquad \{\lambda w.\lambda z.w(\lambda x.\exists e Opened(e) \wedge Opener(e,z) \wedge OpenedThing(e,x))\} \end{array}$$

With this attachment in place, the transitive *VP* rule will incorporate the variable standing for *a restaurant* as the second argument to *opened*, incorporate the entire expression representing the *opening* event as the nuclear scope of *a restaurant* and

Grammar Rule	**Semantic Attachment**
$S \rightarrow NP\ VP$	$\{NP.sem(VP.sem)\}$
$NP \rightarrow Det\ Nominal$	$\{Det.sem(Nominal.sem)\}$
$NP \rightarrow ProperNoun$	$\{ProperNoun.sem\}$
$Nominal \rightarrow Noun$	$\{Noun.sem\}$
$VP \rightarrow Verb$	$\{Verb.sem\}$
$VP \rightarrow Verb\ NP$	$\{Verb.sem(NP.sem)\}$
$Det \rightarrow every$	$\{\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)\}$
$Det \rightarrow a$	$\{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)\}$
$Noun \rightarrow restaurant$	$\{\lambda r.Restaurant(r)\}$
$ProperNoun \rightarrow Matthew$	$\{\lambda m.m(Matthew)\}$
$ProperNoun \rightarrow Franco$	$\{\lambda f.f(Franco)\}$
$ProperNoun \rightarrow Frasca$	$\{\lambda f.f(Frasca)\}$
$Verb \rightarrow closed$	$\{\lambda x.\exists e Closed(e) \wedge ClosedThing(e,x)\}$
$Verb \rightarrow opened$	$\{\lambda w.\lambda z.w(\lambda x.\exists e Opened(e) \wedge Opener(e,z)$ $\wedge Opened(e,x))\}$

Figure 18.3 Semantic attachments for a fragment of our English grammar and lexicon.

finally produce a λ-expression suitable for use with our S rule. As with the previous example you should walk through this example step by step to make sure that you arrive at our intended meaning representation.

$$\exists x Restaurant(x) \wedge \exists e Opened(e) \wedge Opener(e, Matthew) \wedge OpenedThing(e,x)$$

The list of semantic attachments which we've developed for this small grammar fragment is shown in Fig. 18.2. Sec. 18.5 expands the coverage of this fragment to some of the more important constructions in English.

In walking through these examples, we have introduced three techniques that instantiate the rule-to-rule approach to semantic analysis introduced at the beginning of this section:

1. Associating complex, function-like, λ-expressions with lexical items
2. Copying of semantic values from children to parents in non-branching rules
3. Function-like application of the semantics of one of the children of a rule to the semantics of the other children of the rule via λ-reduction.

These techniques serve to illustrate a general division of labor that guides the design of semantic attachments in this compositional framework. In general, it is the lexical rules that introduce quantifiers, predicates and terms into our meaning representations. The semantic attachments for grammar rules put these elements together in the right ways, but do not in general introduce new elements into the representations being created.

18.3 Quantifier Scope Ambiguity and Underspecification

The grammar fragment developed in the last section appears to be sufficient to handle examples like the following that contain two or more quantified noun phrases.

(18.6) Every restaurant has a menu.

Systematically applying the rules given in Fig. 18.2 to this example produces the following perfectly reasonable meaning representation.

$$\forall x\, Restaurant(x) \Rightarrow \\ \exists y\, Menu(y) \wedge \exists e Having(e) \wedge Haver(e,x) \wedge Had(e,y)$$

This formula more or less corresponds to the common sense notion that all restaurants have menus.

Unfortunately, this isn't the only possible interpretation for this example. The following is also possible.

$$\exists y\, Menu(y) \wedge \forall x\, Restaurant(x) \Rightarrow \\ \exists e\, Having(e) \wedge Haver(e,x) \wedge Had(e,y)$$

This formula asserts that there is one menu out there in the world and all restaurants share it. Now from a common sense point of view this seems pretty unlikely, but remember that our semantic analyzer only has access to the semantic attachments in the grammar and the lexicon in producing meaning representations. Of course, world knowledge and contextual information can be used to select between these two readings, but only if we are able to produce both.

This example illustrates that expressions containing quantified terms can give rise to ambiguous representations even in the absence of syntactic, lexical or anaphoric ambiguities. This is known as the problem of **quantifier scoping**. The difference between the two interpretations given above arises from which of the two quantified variables has the outer scope.

Quantifer scoping

The approach outlined in the last section can not handle this phenomena. To fix this we'll need the following capabilities.

- The ability to efficiently create *underspecified* representations that embody all possible readings without explicitly enumerating them
- A means to generate, or extract, all of the possible readings from this representation
- And the ability to choose among the possible readings

The following sections will outline approaches to the first two problems. The solution to the last, most important problem, requires the use of context and world knowledge and unfortunately remains a largely unsolved problem.

18.3.1 Store and Retrieve Approaches

One way to address the quantifier scope problem is to add a new notation to our existing semantic attachments to facilitate the compositional creation of the desired meaning

Complex-term

representations. In this case, we'll introduce the notion of a **complex-term** that permits FOL expressions like $\forall x\, Restaurant(x)$ to appear in places where we would normally only allow FOL terms to appear. Formally, a complex-term will be an expression with the following three-part structure:

$$\langle Quantifier\ variable\ formula \rangle$$

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

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

The intent of the this approach is to capture the basic predicate argument structure of an expression, while remaining agnostic about where the various quantifiers will end up in the final representation.

As was the case with λ-expressions, this notational device is only useful if we can provide an algorithm to convert it back into an ordinary FOL expression. This can be accomplished by rewriting any predicate containing a complex-term according to the following schema:

$$\begin{aligned}&P(\langle Quantifier\ variable\ formula\rangle)\\ &\Longrightarrow\\ &Quantifier\ variable\ formula\ Connective\ P(variable)\end{aligned}$$

In other words, the complex-term:

1. is extracted from the predicate in which it appears,
2. is replaced by the specified variable,
3. and has its variable, quantifier, and formula prepended to the new expression through the use of an appropriate connective.

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 $\forall$.

How does this scheme help with our ambiguity problem? Note that our new representation contains two complex terms. The order in which we process them determines which of the two readings we end up with. Let's consider the case where we proceed left-to-right through the expression transforming the complex terms as we find them. In this case, we encounter *Every restaurant* first; transforming it yields the following expression.

$$\forall x Restaurant(x) \Rightarrow \exists e\, Having(e) \wedge Haver(e,x) \wedge Had(e, \langle \exists y Menu(y)\rangle)$$

Proceeding onward we next encounter *a menu*. Transforming this complex term yields the following final form which corresponds to the non-intuitive reading that we couldn't get with our earlier method.

$$\exists y Menu(y) \wedge \forall x Restaurant(x) \Rightarrow \exists e\, Having(e) \wedge Haver(e,x) \wedge Had(e,y)$$

To get the more common-sense reading that we had earlier all we have to is pull out the complex-terms in the other order; first *a menu* and then *every restaurant*.

This approach to quantifier scope provides solutions to the two of the desiderata given earlier: complex terms provide a compact underspecified representation of all the possible quantifier-based ambiguous readings, and the method for transforming them provides a deterministic method for eliminating complex terms and thus retrieving valid FOL formulas. And by altering the ordering by which complex terms are eliminated we can recover all the possible readings. Of course, sentences with N quantifiers will have $O(N!)$ different quantifier-based readings.

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. Alshawi (1992) presents a comprehensive approach to generating plausible quantifier scopings.

18.4 Unification-Based Approaches to Semantic Analysis

As mentioned in Sec. 18.2, feature structures and the unification operator provide an effective way to implement syntax-driven semantic analysis. Recall that in Ch. 16 we paired complex feature structures with individual context-free grammar rules to encode syntactic constraints such as number agreement and subcategorization; constraints that were awkward or in some cases impossible to convey directly using context-free grammars. For example, the following rule was used to capture agreement constraints on English noun phrases.

$$\begin{aligned}&NP \rightarrow Det\ Nominal\\ &\quad \langle Det\ \textsc{agreement}\rangle = \langle Nominal\ \textsc{agreement}\rangle\\ &\quad \langle NP\ \textsc{agreement}\rangle = \langle Nominal\ \textsc{agreement}\rangle\end{aligned}$$

Rules such as this one serve two functions at the same time: they insure that the grammar rejects expressions that violate this constraint, and more importantly for our current topic, they create complex structures that can be associated with parts of grammatical derivations. The following structure, for example, results from the application of the above rule to a singular noun phrase.

$$\begin{bmatrix}\textsc{agreement} & \begin{bmatrix}\textsc{number} & sg\end{bmatrix}\end{bmatrix}$$

We'll use this latter capability to compose meaning representations and associate them with constituents in parse.

In this unification-based approach, our FOL representations and λ-based semantic attachments are replaced by complex feature structures and unification equations. To see how this works, let's walk through a series of examples similar to those discussed earlier in Sec. 18.2. Let's start with a simple intransitive sentence with a proper noun as it's subject.

(18.7) Rhumba closed

Using an event-oriented approach, the meaning representation for this sentence should be something like the following.

$$\exists e\, Closing(e) \wedge Closed(e, Rhumba)$$

Our first task will be to show that we can encode representations like this within the feature structure framework. The most straightforward way to approach this task is to simply follow the BNF-style definition of FOL statements given in Ch. 17. The relevant elements of this definition stipulate that FOL formulas come in three varieties: atomic formulas consisting of predicates with the appropriate number of term arguments, formulas conjoined with other formulas via the $\wedge$, $\vee$ and $\Rightarrow$ operators, and finally quantified formulas which consist of a quantifier, variables and a formula. Using this definition as a guide, we can capture this FOL expression with the following feature structure.

$$\begin{bmatrix} \text{QUANT} & \exists \\ \text{VAR} & \boxed{1} \\ \text{FORMULA} & \begin{bmatrix} \text{OP} & \text{AND} \\ \text{FORMULA1} & \begin{bmatrix} \text{PRED} & \text{CLOSING} \\ \text{ARG0} & \boxed{1} \end{bmatrix} \\ \text{FORMULA2} & \begin{bmatrix} \text{PRED} & \text{CLOSED} \\ \text{ARG0} & \boxed{1} \\ \text{ARG1} & \text{RHUMBA} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Fig. 18.4 shows this expression using the DAG-style notation introduced in Ch. 16. This figure reveals the way that variables are handled. Instead of introducing explicit FOL variables, we'll use the path-based feature-sharing capability of feature structures to accomplish the same goal. In this example, the event variable *e* is captured by the three paths leading to the same shared node.

Our next step is to associate unification equations with the grammar rules involved in this example's derivation. Let's start at the top with the *S* rule.

$$\begin{aligned} &S \rightarrow NP\ VP \\ &\quad \langle S\ \text{SEM} \rangle = \langle NP\ \text{SEM} \rangle \\ &\quad \langle VP\ \text{ARG0} \rangle = \langle NP\ \text{INDEXVAR} \rangle \\ &\quad \langle NP\ \text{SCOPE} \rangle = \langle VP\ \text{SEM} \rangle \end{aligned}$$

The first line simply equates the meaning representation of the *NP* (encoded under the SEM feature) with our top-level *S*. The purpose of the second equation is to assign the subject *NP* to the appropriate role inside the *VP*'s meaning representation. More concretely, it fills the appropriate role in the *VP*'s semantic representation by unifying the ARG0 feature with a path that leads to a representation of the semantics of the *NP*.

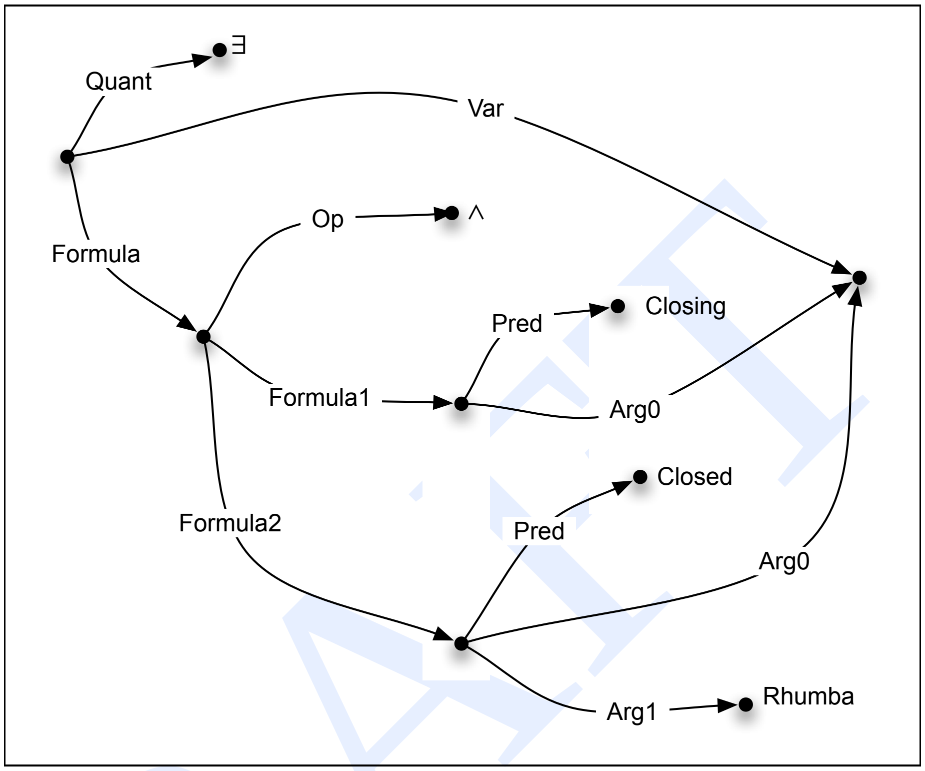

Figure 18.4 A directed graph notation for semantic feature structures.

Finally, it unifies the SCOPE feature in the *NP*'s meaning representation with a pointer to the VP's meaning representation. As we'll see, this is a somewhat convoluted way to bring the representation of an event up to where it belongs in the representation. The motivation for this apparatus should become clear in the ensuing discussion where we consider quantified noun phrases.

Carrying on, let's consider the attachments for the *NP* and *ProperNoun* parts of this derivation.

$$\begin{array}{l} NP \rightarrow ProperNoun \\ \quad \langle NP\ \textsc{sem}\rangle = \langle ProperNoun\ \textsc{sem}\rangle \\ \quad \langle NP\ \textsc{scope}\rangle = \langle \text{ProperNoun}\ \textsc{scope}\rangle \\ \quad \langle NP\ \textsc{indexvar}\rangle = \langle ProperNoun\ \textsc{indexvar}\rangle \end{array}$$

$$\begin{array}{l} ProperNoun \rightarrow Rhumba \\ \quad \langle ProperNoun\ \textsc{sem pred}\rangle = \textsc{rhumba} \\ \quad \langle ProperNoun\ \textsc{indexvar}\rangle = \langle ProperNoun\ \textsc{sem pred}\rangle \end{array}$$

As we saw earlier, there isn't much to the semantics of proper nouns in this approach. Here we're just introducing a constant and providing an index variable to point at that constant.

Next, let's move on to the semantic attachments for the *VP* and *Verb* rules.

$$\begin{aligned}&VP \rightarrow Verb\\ &\quad \langle VP\ \text{SEM}\rangle = \langle\ \text{Verb SEM}\rangle\\ &\quad \langle VP\ \text{ARG0}\rangle = \langle\ \text{Verb ARG0}\rangle\end{aligned}$$

$$\begin{aligned}&Verb \rightarrow closed\\ &\quad \langle Verb\ \text{SEM QUANT}\rangle = \exists\\ &\quad \langle Verb\ \text{SEM FORMULA OP}\rangle = \wedge\\ &\quad \langle Verb\ \text{SEM FORMULA FORMULA1 PRED}\rangle = \text{CLOSING}\\ &\quad \langle Verb\ \text{SEM FORMULA FORMULA1 ARG0}\rangle = \langle Verb\ \text{SEM VAR}\rangle\\ &\quad \langle Verb\ \text{SEM FORMULA FORMULA2 PRED}\rangle = \text{CLOSED}\\ &\quad \langle Verb\ \text{SEM FORMULA FORMULA2 ARG0}\rangle = \langle Verb\ \text{SEM VAR}\rangle\\ &\quad \langle Verb\ \text{SEM FORMULA FORMULA2 ARG1}\rangle = \langle Verb\ \text{ARG0}\rangle\end{aligned}$$

The attachments for the *VP* rule parallel our earlier treatment of non-branching grammatical rules. These unification equations are simply making the appropriate semantic fragments of the *Verb* available at the *VP* level. In contrast, the unification equations for the *Verb* introduce the bulk of the event representation that is at the core of this example. Specifically, it introduces the quantifier, event variable and predications that make up the body of the final expression. What would be an event variable in FOL is captured by the equations unifying the *Verb* SEM VAR path with the appropriate arguments to the predicates in the body of the formula. Finally, it exposes the single missing argument (the entity being closed) through the ⟨ *Verb* ARG0⟩ equation.

Taking a step back we can see that these equations serve the same basic functions as the λ-expressions in Sec. 18.2; they provide the content of the FOL formula being created, and they serve to expose and name the external arguments that will be filled in later at higher levels in the grammar.

These last few rules also display the division of labor that we've seen several times now; lexical rules introduce the bulk of the semantic content, while higher level grammatical rules assemble the pieces in the right way, rather than introducing content.

Of course, as was the case with the λ-based approach things get quite a bit more complex when we look at expressions containing quantifiers. To see this, let's work through the following example.

(18.8) Every restaurant closed

Again, the meaning representation for this expression should be the following

$$\forall x Restaurant(x) \Rightarrow (\exists e Closing(e) \wedge Closed(e,x))$$

which is captured by the following feature structure.

$$\begin{bmatrix} \text{QUANT} & \forall \\ \text{VAR} & \boxed{1} \\ \text{FORMULA} & \begin{bmatrix} \text{OP} & \Rightarrow \\ \text{FORMULA1} & \begin{bmatrix} \text{PRED} & \text{RESTAURANT} \\ \text{ARG0} & \boxed{1} \end{bmatrix} \\ \text{FORMULA2} & \begin{bmatrix} \text{QUANT} & \text{EXISTS} \\ \text{VAR} & \boxed{2} \\ \text{FORMULA} & \begin{bmatrix} \text{OP} & \wedge \\ \text{FORMULA1} & \begin{bmatrix} \text{PRED} & \text{CLOSING} \\ \text{ARG0} & \boxed{2} \end{bmatrix} \\ \text{FORMULA2} & \begin{bmatrix} \text{PRED} & \text{CLOSED} \\ \text{ARG0} & \boxed{2} \\ \text{ARG1} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

As we saw earlier with the λ-based approach, the outer structure for expressions like this comes largely from the subject noun phrase. Recall that schematically this semantic structure has the form $\forall x P(x) \Rightarrow Q(x)$ where the P expression is traditionally referred to as the *restrictor* and is provided by the head noun and Q is referred to as the *nuclear scope* and comes from the verb phrase.

This structure gives rise to two distinct tasks for our semantic attachments: the semantics of the *VP* semantics must be unified with the nuclear scope of the subject noun phrase, and the variable representing that noun phrase must be assigned to the ARG1 role of the CLOSED predicate in the event structure. The following rules involved in the derivation of *Every restaurant* address these two tasks

$$NP \rightarrow Det\ Nominal$$
$$\langle\ NP\ \text{SEM}\rangle = \langle Det\ \text{SEM}\ \rangle$$
$$\langle\ NP\ \text{SEM VAR}\ \rangle = \langle\ NP\ \text{INDEXVAR}\ \rangle$$
$$\langle\ NP\ \text{SEM FORMULA FORMULA1}\ \rangle = \langle\ Nominal\ \text{SEM}\ \rangle$$
$$\langle\ NP\ \text{SEM FORMULA FORMULA2}\ \rangle = \langle\ NP\ \text{SCOPE}\ \rangle$$

$$Nominal \rightarrow Noun$$
$$\langle\ Nominal\ \text{SEM}\ \rangle = \langle\ Noun\ \text{SEM}\ \rangle$$
$$\langle\ Nominal\ \text{INDEXVAR}\ \rangle = \langle\ Noun\ \text{INDEXVAR}\ \rangle$$

$$Noun \rightarrow restaurant$$
$$\langle\ Noun\ \text{SEM PRED}\ \rangle = \langle\ \text{RESTAURANT}\ \rangle$$
$$\langle\ Noun\ \text{INDEXVAR}\ \rangle = \langle\ Noun\ \text{SEM PRED}\ \rangle$$

$$Det \rightarrow every$$

$$\langle\ \textit{Det}\ \text{SEM QUANT}\ \rangle = \forall$$
$$\langle\ \textit{Det}\ \text{SEM FORMULA OP}\ \rangle = \ \Rightarrow$$

As one final exercise, let's walk through an example with a transitive verb phrase.

(18.9) Franco opened a restaurant

This example has the following meaning representation.

$$\exists x\, Restaurant(x) \wedge \exists e\, Opening(e) \wedge Opener(e, Franco) \wedge Opened(e, x)$$

$$\left[\begin{array}{ll}
\text{QUANT} & \text{EXISTS}\\
\text{VAR} & \boxed{1}\\
\text{FORMULA} & \left[\begin{array}{ll}
\text{OP} & \wedge\\
\text{FORMULA1} & \left[\begin{array}{ll}\text{PRED} & \text{RESTAURANT}\\ \text{ARG1} & \boxed{1}\end{array}\right]\\
\text{FORMULA2} & \left[\begin{array}{ll}
\text{QUANT} & \exists\\
\text{VAR} & \boxed{2}\\
\text{FORMULA} & \left[\begin{array}{ll}
\text{OP} & \wedge\\
\text{FORMULA1} & \left[\begin{array}{ll}\text{PRED} & \text{OPENING}\\ \text{ARG0} & \boxed{2}\end{array}\right]\\
\text{FORMULA2} & \left[\begin{array}{ll}\text{PRED} & \text{OPENER}\\ \text{ARG0} & \boxed{2}\\ \text{ARG1} & \text{FRANCO}\end{array}\right]\\
\text{FORMULA3} & \left[\begin{array}{ll}\text{PRED} & \text{OPENED}\\ \text{ARG0} & \boxed{2}\\ \text{ARG1} & \boxed{1}\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]$$

The only really new element that we need to address in this example is the following transitive *VP* rule.

$$VP \rightarrow Verb\ NP$$
$$\langle VP\ \text{SEM}\rangle = \langle Verb\ \text{SEM}\rangle$$
$$\langle NP\ \text{SCOPE}\rangle = \langle VP\ \text{SEM}\rangle$$
$$\langle Verb\ \text{ARG1}\rangle = \langle NP\ \text{INDEXVAR}\rangle$$

This rule has the two primary tasks that parallel those in our *S* rule: it has to fill the nuclear scope of the object *NP* with the semantics of the *VP*, and it has to insert the variable representing the object into to the right role in the *VP*'s meaning representation.

One obvious problem with the approach we just described is that it fails to generate all the possible ambiguous representations arising from quantifier scope ambiguities. Fortunately, the approaches to underspecification described earlier in Sec. 18.3 can be adapted to the unification-based approach.

18.5 Semantic Attachments for a Fragment of English

This section describes a set of semantic attachments for a small fragment of English, the bulk of which are based on those used in the Core Language Engine (Alshawi, 1992). 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.

18.5.1 Sentences

To this point, we've only dealt with simple declarative sentences. This section expands our coverage to include the other sentence types first introduced in Ch. 12: imperatives, yes-no-questions, and wh-questions. Let's start by considering the following examples:

(18.10) Flight 487 serves lunch.
(18.11) Serve lunch.
(18.12) Does Flight 207 serve lunch?
(18.13) 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 listener, 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 FOL sentences in the same way that belief operators were used in Ch. 17. Specifically, the operators *DCL*, *IMP*, *YNQ*, and *WHQ* will be applied to the FOL 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:

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

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:

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

Applying this rule to (18.11), results in the following representation:

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

As will be discussed in Ch. 24, imperatives can be viewed as a kind of **speech act**.

As discussed in Ch. 12, **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:

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

The use of this rule with for example (18.12) 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 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 Ch. 17, yes-no-questions can be answered by determining if the proposition is in the knowledge-base, or can be inferred from it.

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:

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

The following representation is the result of applying this rule to example (18.13):

$$\begin{aligned}WHQ(x, \exists e, x\ &Serving(e) \wedge Server(e, x)\\ &\wedge Served(e, Lunch) \wedge Flight(x))\end{aligned}$$

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**:

(18.14) 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:

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

The following representation would result from an application of this rule to example (18.14):

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

As we'll see in Ch. 24, correctly answering this kind of question involves a fair amount of domain specific reasoning. For example, the correct way to answer example (18.14) 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 Ch. 16, 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.

18.5.2 Noun Phrases

As we have already seen, the meaning representations for noun phrases can be either normal FOL 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:

(18.15) Flight schedule
(18.16) Summer flight schedule

As noted in Ch. 12, the syntactic structure of this construction can be captured by the regular expression *Noun*∗, or by the following context-free grammar rules:

$$\begin{aligned} \mathit{Nominal} &\rightarrow \mathit{Noun} \\ \mathit{Nominal} &\rightarrow \mathit{Nominal\ Noun} \end{aligned}$$

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:

$$\begin{array}{l} \textit{Nominal} \rightarrow \textit{Nominal Noun} \\ \quad \{\lambda x\ \textit{Nominal.sem}(x) \wedge \textit{NN}(\textit{Noun.sem}, x)\} \end{array}$$

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:

$$\lambda x.Schedule(x) \wedge NN(x, Flight)$$

$$\lambda x.Schedule(x) \wedge NN(x, Flight) \wedge NN(x, Summer)$$

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 Ch. 12 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's best to simply state an abstract semantic relation between the various constituents.

$$\begin{array}{l} \textit{NP} \rightarrow \textit{ComplexDet Nominal} \\ \quad \{\langle \exists x Nominal.sem(x) \wedge GN(x, ComplexDet.sem)\rangle\} \end{array}$$

$$\textit{ComplexDet} \rightarrow \textit{NP 's} \qquad \{\textit{NP.sem}\}$$

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 predicative. These categories are exemplified by the following BERP examples:

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

(18.18) 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:

$$Nominal \rightarrow Adj\ Nominal$$
$$\{\lambda x\, Nominal.sem(x) \wedge Isa(x, Adj.sem)\}$$

$$Adj \rightarrow cheap \qquad \{Cheap\}$$

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 not unreasonable representation:

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

Intersective semantics

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*:

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

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

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

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

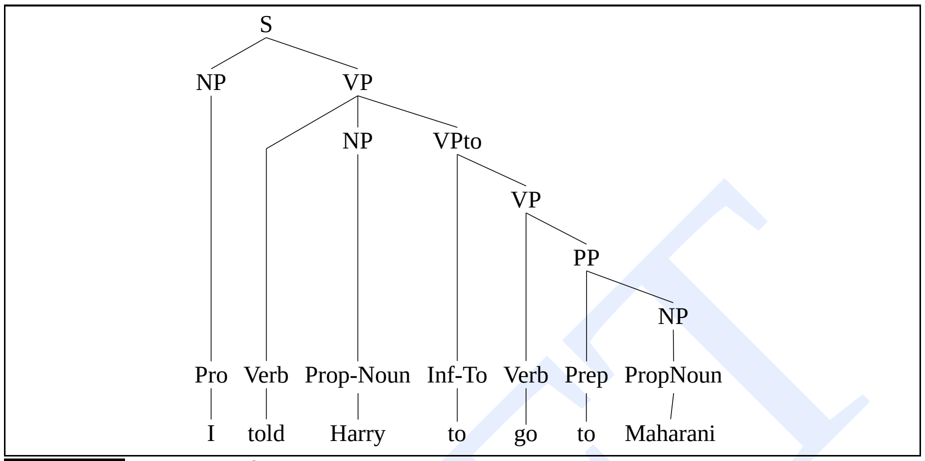

Figure 18.5 Parse tree for *I told Harry to go to Maharani.*

further procedure with access to additional relevant knowledge can replace this vague relation with an appropriate representation (Alshawi, 1992).

$$\begin{array}{l} \textit{Nominal} \rightarrow \textit{Adj Nominal} \\ \qquad \{\lambda x\, Nominal.sem(x) \wedge AM(x, Adj.sem)\} \end{array}$$

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

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

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.

18.5.3 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.

Consider the following example:

(18.19) I told Harry to go to Maharani.

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

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

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 need something like the following as its semantic attachment:

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

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 18.5, *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:

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

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 be used with this construction.

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

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:

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

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 FOL 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.

18.5.4 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:

(18.20) A restaurant on Broadway.

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

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

$$NP \rightarrow Det\ Nominal$$
$$Nominal \rightarrow Nominal\ PP$$
$$PP \rightarrow P\ NP$$

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:

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

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:

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

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:

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

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.

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

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

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

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:

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

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 notable aspect of 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 to 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:

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

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:

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

In examples like this, the arguments of *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.

$$PP \rightarrow P\ NP \qquad \{NP.sem\}$$

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 Ch. 19.

18.6 Integrating Semantics into the Earley Parser

In Section 18.1, 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 Ch. 13.

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 Ch. 16. Three modifications are required to the original algorithm:

1. The rules of the grammar are given a new field to contain their semantic attachments.

2. The states in the chart are given a new field to hold the meaning representation of the constituent.
3. 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 18.6 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 18.6 shows ENQUEUE 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. Ch. 19 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.

18.7 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.

Voltaire[2], 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:

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

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

(18.24) 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 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:

$$\begin{array}{l} NP \rightarrow \textit{the tip of the iceberg} \\ \qquad \{Beginning\} \end{array}$$

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.

As with the rest of the grammar, it may take a few tries to get these rules right. Consider the following *iceberg* examples from the WSJ corpus:

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

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

(18.27) 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 idiom that permits

2 *Essai sur les moeurs et les esprit des nations.* Translation by Y. Sills, as quoted in Sills and Merton (1991).

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 like the following:

$$\begin{aligned} NP &\rightarrow \textit{TipNP of IcebergNP} \\ &\quad \{\textit{Beginning}\} \end{aligned}$$

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 Ch. 19, 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 Ch. 25, machine translation.

18.8 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 of an input.
- Compositional creation of FOL formulas is possible with a few notational extensions including **λ-expressions** and **complex-terms**.
- Compositional creation of FOL formulas is also possible using the mechanisms provided by feature structures and unification.

- **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.

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 Ch. 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 18.5 are adapted from SRI's Core Language Engine (Alshawi, 1992). Additional bits and pieces were adapted from Woods (1977), Schubert and Pelletier (1982), and Gazdar et al. (1985). Of necessity, a large number of important topics were not covered in this chapter. See Alshawi (1992) for the standard gap-threading approach to semantic interpretation in the presence of long-distance dependencies. ter Meulen (1995) presents an modern 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) and Alshawi (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 compound nominals. Linguistic research on this topic can be found in Lees (1970), Downing (1977), Levi (1978), and Ryder (1994), more computational approaches are described in Gershman (1977), Finin (1980), McDonald (1982), Pierre (1984), Arens et al. (1987), Wu (1992), Vanderwende (1994), and Lauer (1995).

There is a long and extensive literature on idioms. Fillmore et al. (1988) describe a general grammatical framework called Construction Grammar 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 in their PHRAN system. 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).

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. Some of the earliest and most successful efforts along these lines were developed by Simmons (1973, 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

18.1 The attachment given on page 609 for handling 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.

18.2 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.

18.3 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 Ch. 17. 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.

18.4 As noted in Ch. 17, 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:

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

18.5 Implement an Earley-style semantic analyzer based on the discussion on page 615.

18.6 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?

18.7 Add a simple type-driven semantics mechanism to the Earley analyzer you implemented for Exercise 5.

18.8 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 (i.e., don't just search for "the tip of the iceberg".) Analyze these examples and come up with a set of grammar rules that correctly accounts for them.

18.9 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.

18.10 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 the 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?