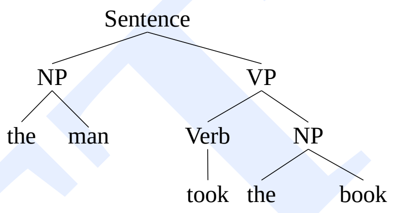
Chapter 12
Formal Grammars of English



The first context-free grammar parse tree (Chomsky, 1956)

If on a winter's night a traveler by Italo Calvino
Nuclear and Radiochemistry by Gerhart Friedlander et al.
The Fire Next Time by James Baldwin
A Tad Overweight, but Violet Eyes to Die For by G. B. Trudeau
Sometimes a Great Notion by Ken Kesey
Dancer from the Dance by Andrew Holleran

Six books in English whose titles are not constituents, from Pullum (1991, p. 195)

The study of grammar has an ancient pedigree; Panini's grammar of Sanskrit was written over two thousand years ago, and is still referenced today in teaching Sanskrit. By contrast, Geoff Pullum noted in a recent talk that "almost everything most educated Americans believe about English grammar is wrong". In this chapter we make a preliminary stab at addressing some of these gaps in our knowledge of grammar and syntax, as well as introducing some of the formal mechanisms that are available for capturing this knowledge.

Syntax

The word **syntax** comes from the Greek *sýntaxis*, meaning "setting out together or arrangement", and refers to the way words are arranged together. We have seen various syntactic notions in previous chapters. The regular languages introduced in Ch. 2 offered a simple way to represent the ordering of strings of words, and Ch. 4 showed how to compute probabilities for these word sequences. Ch. 5 showed that part-of-speech categories could act as a kind of equivalence class for words. This chapter and the following ones introduce sophisticated notions of syntax and grammar that go well beyond these simpler notions. In this chapter, we introduce three main new ideas: **constituency**, **grammatical relations**, and **subcategorization and dependency**.

The fundamental idea of constituency is that groups of words may behave as a single unit or phrase, called a constituent. For example we will see that a group of words called a **noun phrase** often acts as a unit; noun phrases include single words like *she* or *Michael* and phrases like *the house, Russian Hill*, and *a well-weathered* *three-story structure*. This chapter will introduce the use of **context-free grammars**, a formalism that will allow us to model these constituency facts.

Grammatical relations are a formalization of ideas from traditional grammar such as SUBJECTS and OBJECTS, and other related notions. In the following sentence the noun phrase *She* is the SUBJECT and *a mammoth breakfast* is the OBJECT:

(12.1) She ate a mammoth breakfast.

Subcategorization and **dependency relations** refer to certain kinds of relations between words and phrases. For example the verb *want* can be followed by an infinitive, as in *I want to fly to Detroit*, or a noun phrase, as in *I want a flight to Detroit*. But the verb *find* cannot be followed by an infinitive (**I found to fly to Dallas*). These are called facts about the *subcategorization* of the verb.

As we'll see, none of the syntactic mechanisms that we've discussed up until now can easily capture such phenomena. They can be modeled much more naturally by grammars that are based on context-free grammars. Context-free grammars are thus the backbone of many formal models of the syntax of natural language (and, for that matter, of computer languages). As such they are integral to many computational applications including grammar checking, semantic interpretation, dialogue understanding and machine translation. They are powerful enough to express sophisticated relations among the words in a sentence, yet computationally tractable enough that efficient algorithms exist for parsing sentences with them (as we will see in Ch. 13). Later in Ch. 14 we'll show that adding probability to context-free grammars gives us a model of disambiguation, and also helps model certain aspects of human parsing.

In addition to an introduction to the grammar formalism, this chapter also provides a brief overview of the grammar of English. We have chosen a domain which has relatively simple sentences, the Air Traffic Information System (ATIS) domain (Hemphill et al., 1990). ATIS systems are an early example of spoken language systems for helping book airline reservations. Users try to book flights by conversing with the system, specifying constraints like *I'd like to fly from Atlanta to Denver*. The U.S. government funded a number of different research sites to collect data and build ATIS systems in the early 1990s. The sentences we will be modeling in this chapter are drawn from the corpus of user queries to the system.

12.1 Constituency

Noun phrase

How do words group together in English? Consider the **noun phrase**, a sequence of words surrounding at least one noun. Here are some examples of noun phrases (thanks to Damon Runyon):

Harry the Horse	a high-class spot such as Mindy's
the Broadway coppers	the reason he comes into the Hot Box
they	three parties from Brooklyn

How do we know that these words group together (or "form constituents")? One piece of evidence is that they can all appear in similar syntactic environments, for example before a verb.

three parties from Brooklyn *arrive*...
a high-class spot such as Mindy's *attracts*...
the Broadway coppers *love*...
they *sit*

But while the whole noun phrase can occur before a verb, this is not true of each of the individual words that make up a noun phrase. The following are not grammatical sentences of English (recall that we use an asterisk (*) to mark fragments that are not grammatical English sentences):

*from *arrive*...	*as *attracts*...
*the *is*...	*spot *is*...

Thus to correctly describe facts about the ordering of these words in English, we must be able to say things like "*Noun Phrases can occur before verbs*".

Preposed
Postposed

Other kinds of evidence for constituency come from what are called **preposed** or **postposed** constructions. For example, the prepositional phrase *on September seventeenth* can be placed in a number of different locations in the following examples, including preposed at the beginning, and postposed at the end:

On September seventeenth, I'd like to fly from Atlanta to Denver
I'd like to fly *on September seventeenth* from Atlanta to Denver
I'd like to fly from Atlanta to Denver *on September seventeenth*

But again, while the entire phrase can be placed differently, the individual words making up the phrase cannot be:

*On September, I'd like to fly seventeenth from Atlanta to Denver
*On I'd like to fly September seventeenth from Atlanta to Denver
*I'd like to fly on September from Atlanta to Denver seventeenth

Section 12.6 will give other motivations for context-free grammars based on their ability to model recursive structures. See Radford (1988) for further examples of groups of words behaving as a single constituent.

12.2 Context-Free Grammars

CFG

The most commonly used mathematical system for modeling constituent structure in English and other natural languages is the **Context-Free Grammar**, or **CFG** Context-free grammars are also called **Phrase-Structure Grammars**, and the formalism is equivalent to what is also called **Backus-Naur Form** or **BNF**. The idea of basing a grammar on constituent structure dates back to the psychologist Wilhelm Wundt (1900), but was not formalized until Chomsky (1956) and, independently, Backus (1959).

Rules
Lexicon

A context-free grammar consists of a set of **rules** or **productions**, each of which expresses the ways that symbols of the language can be grouped and ordered together, and a **lexicon** of words and symbols. For example, the following productions express

NP

that a **NP** (or **noun phrase**), can be composed of either a *ProperNoun* or a determiner (*Det*) followed by a *Nominal*; a *Nominal* can be one or more *Noun*s.

$$\begin{aligned} NP &\rightarrow Det\ Nominal \\ NP &\rightarrow ProperNoun \\ Nominal &\rightarrow Noun \mid Nominal\ Noun \end{aligned}$$

Context-free rules can be hierarchically embedded, so we can combine the previous rules with others like the following which express facts about the lexicon:

$$\begin{aligned} Det &\rightarrow a \\ Det &\rightarrow the \\ Noun &\rightarrow flight \end{aligned}$$

Terminal

Non-terminal

The symbols that are used in a CFG are divided into two classes. The symbols that correspond to words in the language ("the", "nightclub") are called **terminal** symbols; the lexicon is the set of rules that introduce these terminal symbols. The symbols that express clusters or generalizations of these are called **non-terminals**. In each context-free rule, the item to the right of the arrow (→) is an ordered list of one or more terminals and non-terminals, while to the left of the arrow is a single non-terminal symbol expressing some cluster or generalization. Notice that in the lexicon, the non-terminal associated with each word is its lexical category, or part-of-speech, which we defined in Ch. 5.

A CFG can be thought of in two ways: as a device for generating sentences, and as a device for assigning a structure to a given sentence. We saw this same dualism in our discussion of finite-state transducers in Ch. 3. As a generator, we can read the → arrow as "rewrite the symbol on the left with the string of symbols on the right".

So starting from the symbol:	*NP*,
we can use rule 12.2 to rewrite *NP* as:	*Det Nominal*
and then rule 12.2:	*Det Noun*
and finally via rules 12.2 and 12.2 as:	*a flight*

Derivation

Parse tree

We say the string *a flight* can be derived from the non-terminal *NP*. Thus a CFG can be used to generate a set of strings. This sequence of rule expansions is called a **derivation** of the string of words. It is common to represent a derivation by a **parse tree** (commonly shown inverted with the root at the top). Fig. 12.1 shows the tree representation of this derivation.

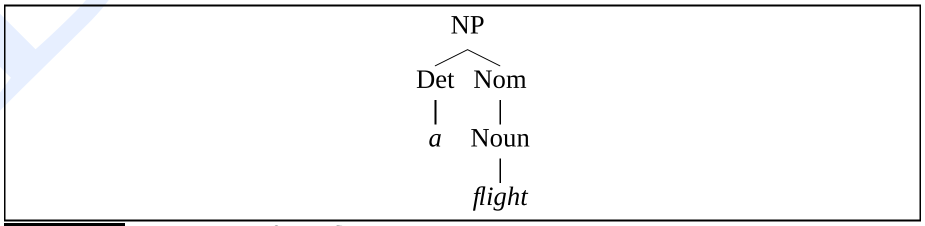


Figure 12.1 A parse tree for "a flight".

Dominates

In the parse tree shown in Fig. 12.1, we can say that the node *NP* **dominates** all the nodes in the tree (*Det*, *Nom*, *Noun*, *a*, *flight*). We can say further that it immediately dominates the nodes *Det* and *Nom*.

Start symbol

The formal language defined by a CFG is the set of strings that are derivable from the designated **start symbol**. Each grammar must have one designated start symbol, which is often called *S*. Since context-free grammars are often used to define sentences, *S* is usually interpreted as the "sentence" node, and the set of strings that are derivable from *S* is the set of sentences in some simplified version of English.

Verb phrase

Let's add to our list of rules a few higher-level rules that expand *S*, and a couple of others. One will express the fact that a sentence can consist of a noun phrase followed by a **verb phrase**:

$$S \rightarrow NP \; VP \quad \text{I prefer a morning flight}$$

A verb phrase in English consists of a verb followed by assorted other things; for example, one kind of verb phrase consists of a verb followed by a noun phrase:

$$VP \rightarrow Verb \; NP \quad \text{prefer a morning flight}$$

Or the verb phrase may have a verb followed by a noun phrase and a prepositional phrase:

$$VP \rightarrow Verb \; NP \; PP \quad \text{leave Boston in the morning}$$

Or the verb may be followed by a prepositional phrase alone:

$$VP \rightarrow Verb \; PP \quad \text{leaving on Thursday}$$

A prepositional phrase generally has a preposition followed by a noun phrase. For example, a very common type of prepositional phrase in the ATIS corpus is used to indicate location or direction:

$$PP \rightarrow Preposition \; NP \quad \text{from Los Angeles}$$

The *NP* inside a *PP* need not be a location; *PPs* are often used with times and dates, and with other nouns as well; they can be arbitrarily complex. Here are ten examples from the ATIS corpus:

to Seattle	on these flights
in Minneapolis	about the ground transportation in Chicago
on Wednesday	of the round trip flight on United Airlines
in the evening	of the AP fifty seven flight
on the ninth of July	with a stopover in Nashville

Fig. 12.2 gives a sample lexicon and Fig. 12.3 summarizes the grammar rules we've seen so far, which we'll call $\mathscr{L}_0$. Note that we can use the or-symbol | to indicate that a non-terminal has alternate possible expansions.

We can use this grammar to generate sentences of this "ATIS-language". We start with *S*, expand it to *NP VP*, then choose a random expansion of *NP* (let's say to *I*), and a random expansion of *VP* (let's say to *Verb NP*), and so on until we generate the

Noun →	*flights* | *breeze* | *trip* | *morning*
Verb →	*is* | *prefer* | *like* | *need* | *want* | *fly*
Adjective →	*cheapest* | *non – stop* | *first* | *latest*
	| *other* | *direct*
Pronoun →	*me* | *I* | *you* | *it*
Proper-Noun →	*Alaska* | *Baltimore* | *Los Angeles*
	| *Chicago* | *United* | *American*
Determiner →	*the* | *a* | *an* | *this* | *these* | *that*
Preposition →	*from* | *to* | *on* | *near*
Conjunction →	*and* | *or* | *but*

Figure 12.2 The lexicon for $\mathscr{L}_0$.

Grammar Rules			**Examples**
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles

Figure 12.3 The grammar for $\mathscr{L}_0$, with example phrases for each rule.

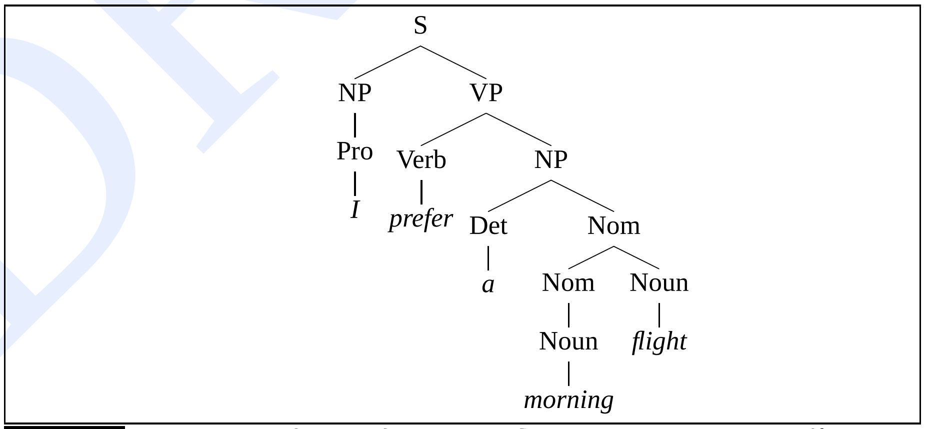


Figure 12.4 The parse tree for "I prefer a morning flight" according to grammar $\mathscr{L}_0$.

string *I prefer a morning flight*. Fig. 12.4 shows a parse tree that represents a complete derivation of *I prefer a morning flight*.

Bracketed notation

It is sometimes convenient to represent a parse tree in a more compact format called **bracketed notation**, essentially the same as LISP tree representations; here is the bracketed representation of the parse tree of Fig. 12.4:

(12.2) $[_S\ [_{NP}\ [_{Pro}$ I]] $[_{VP}\ [_V$ prefer] $[_{NP}\ [_{Det}$ a] $[_{Nom}\ [_N$ morning] $[_{Nom}\ [_N$ flight]]]]]]

Grammatical

Ungrammatical

Generative grammar

A CFG like that of $\mathscr{L}_0$ defines a formal language. We saw in Ch. 2 that a formal language is a set of strings. Sentences (strings of words) that can be derived by a grammar are in the formal language defined by that grammar, and are called **grammatical** sentences. Sentences that cannot be derived by a given formal grammar are not in the language defined by that grammar, and are referred to as **ungrammatical**. This hard line between "in" and "out" characterizes all formal languages but is only a very simplified model of how natural languages really work. This is because determining whether a given sentence is part of a given natural language (say English) often depends on the context. In linguistics, the use of formal languages to model natural languages is called **generative grammar**, since the language is defined by the set of possible sentences "generated" by the grammar.

12.2.1 Formal definition of context-free grammar

We conclude this section by way of summary with a quick formal description of a context-free grammar and the language it generates. A context-free grammar G is defined by four parameters N, Σ, R, S (technically "is a 4-tuple"):

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta$,
	where A is a non-terminal,
	β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$
S	a designated **start symbol**

For the remainder of the book we'll adhere to the following conventions when discussing the formal properties (as opposed to explaining particular facts about English or other languages) of context-free grammars.

Capital letters like A, B, and S	Non-terminals
S	The start symbol
Lower-case Greek letters like α, β, and γ	Strings drawn from $(\Sigma \cup N)*$
Lower-case Roman letters like u, v, and w	Strings of terminals

A language is defined via the concept of derivation. One string derives another one if it can be rewritten as the second one via some series of rule applications. More formally, following Hopcroft and Ullman (1979),

Directly derives

if $A \rightarrow \beta$ is a production of P and α and γ are any strings in the set $(\Sigma \cup N)*$, then we say that $\alpha A \gamma$ **directly derives** $\alpha\beta\gamma$, or $\alpha A \gamma \Rightarrow \alpha\beta\gamma$.

Derivation is then a generalization of direct derivation:

Let $\alpha_1, \alpha_2, \ldots, \alpha_m$ be strings in $(\Sigma \cup N)*, m \geq 1$, such that

(12.3) $$\alpha_1 \Rightarrow \alpha_2, \alpha_2 \Rightarrow \alpha_3, \ldots, \alpha_{m-1} \Rightarrow \alpha_m$$

Derives

We say that α_1 **derives** α_m, or $\alpha_1 \overset{*}{\Rightarrow} \alpha_m$.

We can then formally define the language $\mathscr{L}_G$ generated by a grammar G as the set of strings composed of terminal symbols which can be derived from the designated start symbol S.

(12.4) $$\mathscr{L}_G = \{w | w \text{ is in } \Sigma* \text{ and } S \overset{*}{\Rightarrow} w\}$$

Syntactic parsing

The problem of mapping from a string of words to its parse tree is called **syntactic parsing**; we will define algorithms for parsing in Ch. 13.

12.3 Some Grammar Rules for English

In this section we introduce a few more aspects of the phrase structure of English; for consistency we will continue to focus on sentences from the ATIS domain. Because of space limitations, our discussion will necessarily be limited to highlights. Readers are strongly advised to consult a good reference grammar of English, such as Huddleston and Pullum (2002).

12.3.1 Sentence-Level Constructions

In the small grammar $\mathscr{L}_0$, we provided only one sentence-level construction for declarative sentences like *I prefer a morning flight*. There are a large number of constructions for English sentences, but four are particularly common and important: declarative structure, imperative structure, yes-no-question structure, and wh-question structure.

Declarative

Sentences with **declarative** structure have a subject noun phrase followed by a verb phrase, like "I prefer a morning flight". Sentences with this structure have a great number of different uses that we will follow up on in Ch. 24. Here are a number of examples from the ATIS domain:

The flight should be eleven a.m. tomorrow
The return flight should leave at around seven p.m.
I'd like to fly the coach discount class
I want a flight from Ontario to Chicago
I plan to leave on July first around six thirty in the evening

Imperative

Sentences with **imperative** structure often begin with a verb phrase, and have no subject. They are called imperative because they are almost always used for commands and suggestions; in the ATIS domain they are commands to the system.

Show the lowest fare
Show me the cheapest fare that has lunch
Give me Sunday's flights arriving in Las Vegas from New York City

List all flights between five and seven p.m.
Show me all flights that depart before ten a.m. and have first class fares
Please list the flights from Charlotte to Long Beach arriving after lunch time
Show me the last flight to leave

We can model this sentence structure with another rule for the expansion of *S*:

$$S \rightarrow VP$$

Yes-no question

Sentences with **yes-no question** structure are often (though not always) used to ask questions (hence the name), and begin with an auxiliary verb, followed by a subject *NP*, followed by a *VP*. Here are some examples (note that the third example is not really a question but a command or suggestion; Ch. 24 will discuss the uses of these question forms to perform different **pragmatic** functions such as asking, requesting, or suggesting.)

Do any of these flights have stops?
Does American's flight eighteen twenty five serve dinner?
Can you give me the same information for United?

Here's the rule:

$$S \rightarrow Aux\ NP\ VP$$

Wh-phrase
Wh-word

The most complex of the sentence-level structures we will examine are the various **wh-** structures. These are so named because one of their constituents is a **wh-phrase**, that is, one that includes a **wh-word** (*who, whose, when, where, what, which, how, why*). These may be broadly grouped into two classes of sentence-level structures. The **wh-subject-question** structure is identical to the declarative structure, except that the first noun phrase contains some wh-word.

What airlines fly from Burbank to Denver?
Which flights depart Burbank after noon and arrive in Denver by six p.m?
Whose flights serve breakfast?
Which of these flights have the longest layover in Nashville?

Here is a rule. Exercise 10 discusses rules for the constituents that make up the *Wh-NP*.

$$S \rightarrow Wh\text{-}NP\ VP$$

Wh-non-subject question

In the **wh-non-subject question** structure, the wh-phrase is not the subject of the sentence, and so the sentence includes another subject. In these types of sentences the auxiliary appears before the subject *NP*, just as in the yes-no-question structures. Here is an example followed by a sample rule:

What flights do you have from Burbank to Tacoma Washington?

$$S \rightarrow Wh\text{-}NP\ Aux\ NP\ VP$$

Long-distance dependencies

Constructions like the **wh-non-subject-question** contain what are called **long-distance dependencies** because the *Wh-NP what flights* is far away from the predicate that it is semantically related to, the main verb *have* in the *VP*. In some models of parsing and understanding compatible with the grammar rule above, long-distance dependencies like the relation between *flights* and *have* are thought of as a semantic relation. In such models, the job of figuring out that *flights* is the argument of *have* is done during semantic interpretation. In other models of parsing, the relationship between *flights* and *have* is considered to be a syntactic relation, and the grammar is modified to insert a small marker called a **trace** or **empty category** after the verb. We'll return to such empty-category models when we introduce the Penn Treebank on page 408.

There are other sentence-level structures we won't try to model here, like **topicalization** or other fronting constructions. In topicalization (also treated as a long-distance dependency in the Penn Treebank), a phrase is placed at the beginning of the sentence for discourse purposes.

On Tuesday, I'd like to fly from Detroit to Saint Petersburg

12.3.2 Clauses and Sentences

Before we move on, we should clarify the status of the *S* rules in the grammars we just described. *S* rules are intended to account for entire sentences that stand alone as fundamental units of discourse. However, as we'll see, *S* can also occur on the right-hand side of grammar rules and hence can be embedded within larger sentences. Clearly then there's more to being an *S* then just standing alone as a unit of discourse.

What differentiates sentence constructions (i.e., the *S* rules) from the rest of the grammar is the notion that they are in some sense *complete*. In this way they correspond to the notion of a **clause** in traditional grammars, which are often described as forming a complete thought. One way of making this notion of 'complete thought' more precise is to say an *S* is a node of the parse tree below which the main verb of the *S* has all of its **arguments**. We'll define verbal arguments later, but for now let's just see an illustration from the tree for *I prefer a morning flight* in Fig. 12.4. The verb *prefer* has two arguments: the subject *I* and the object *a morning flight*. One of the arguments appears below the *VP* node, but the other one, the subject *NP*, appears only below the *S* node.

Clause

12.3.3 The Noun Phrase

Our $\mathscr{L}_0$ grammar introduced three of the most frequent types of noun phrases that occur in English: pronouns, proper-nouns and the *NP* $\rightarrow$ *Det Nominal* construction. While pronouns and proper-nouns can be complex in their own ways, the central focus of this section is on the last type since that is where the bulk of the syntactic complexity resides. We can view these noun phrases consisting of a head, the central noun in the noun phrase, along with various modifiers that can occur before or after the head noun. Let's take a close look at the various parts.

The Determiner

Noun phrases can begin with simple lexical determiners, as in the following examples:

a stop	the flights	this flight
those flights	any flights	some flights

The role of the determiner in English noun phrases can also be filled by more complex expressions, as follows:

United's flight
United's pilot's union
Denver's mayor's mother's canceled flight

In these examples, the role of the determiner is filled by a possessive expression consisting of a noun phrase followed by an *'s* as a possessive marker, as in the following rule.

$$Det \rightarrow NP\ 's$$

The fact that this rule is recursive (since an *NP* can start with a *Det*), will help us model the latter two examples above, where a sequence of possessive expressions serves as a determiner.

There are also circumstances under which determiners are optional in English. For example, determiners may be omitted if the noun they modify is plural:

(12.5) Show me *flights* from San Francisco to Denver on weekdays

As we saw in Ch. 5, **mass nouns** also don't require determination. Recall that mass nouns often (not always) involve something that is treated like a substance (including e.g., *water* and *snow*), don't take the indefinite article "*a*", and don't tend to pluralize. Many abstract nouns are mass nouns (*music*, *homework*). Mass nouns in the ATIS domain include *breakfast*, *lunch*, and *dinner*:

(12.6) Does this flight serve dinner?

Exercise 4 asks the reader to represent this fact in the CFG formalism.

The Nominal

The nominal construction follows the determiner and contains any pre- and post-head noun modifiers. As indicated in grammar $\mathscr{L}_0$, in its simplest form a nominal can consist of a single noun.

$$Nominal \rightarrow Noun$$

As we'll see, this rule also provides the basis for the bottom of various recursive rules used to capture more complex nominal constructions.

Before the Head Noun

Cardinal numbers
Ordinal numbers
Quantifiers

A number of different kinds of word classes can appear before the head noun (the "postdeterminers") in a nominal. These include **cardinal numbers**, **ordinal numbers**, and **quantifiers**. Examples of cardinal numbers:

two friends   one stop

Ordinal numbers include *first*, *second*, *third*, and so on, but also words like *next*, *last*, *past*, *other*, and *another*:

the first one	the next day	the second leg
the last flight	the other American flight	

Some quantifiers (*many*, *(a) few*, *several*) occur only with plural count nouns:

many fares

The quantifiers *much* and *a little* occur only with noncount nouns.
Adjectives occur after quantifiers but before nouns.

a *first-class* fare	a *nonstop* flight
the *longest* layover	the *earliest* lunch flight

Adjective phrase

Adjectives can also be grouped into a phrase called an **adjective phrase** or AP. APs can have an adverb before the adjective (see Ch. 5 for definitions of adjectives and adverbs):

the *least expensive* fare

We can combine all the options for prenominal modifiers with one rule as follows:

$$NP \rightarrow (Det)\ (Card)\ (Ord)\ (Quant)\ (AP)\ Nominal$$

This simplified noun phrase rule has a flatter structure and hence is simpler than would be assumed by most modern generative theories of grammar; as we will see in Sec. 12.4, flat structures are often used for simplicity in computational applications (and indeed, there is no universally agreed-upon internal constituency for the noun phrase).

Note the use of parentheses "( )" to mark **optional constituents**. A rule with one set of parentheses is really a shorthand for two rules, one with the parentheses, one without.

After the Head Noun

A head noun can be followed by **postmodifiers**. Three kinds of nominal postmodifiers are very common in English:

prepositional phrases	all flights *from Cleveland*
non-finite clauses	any flights *arriving after eleven a.m.*
relative clauses	a flight *that serves breakfast*

Prepositional phrase postmodifiers are particularly common in the ATIS corpus, since they are used to mark the origin and destination of flights. Here are some examples, with brackets inserted to show the boundaries of each PP; note that more than one PP can be strung together:

any stopovers *[for Delta seven fifty one]*
all flights *[from Cleveland] [to Newark]*
arrival *[in San Jose] [before seven p.m.]*
a reservation *[on flight six oh six] [from Tampa] [to Montreal]*

Here's a new nominal rule to account for postnominal *PP*s:

$$Nominal \rightarrow Nominal\ PP$$

Non-finite

The three most common kinds of **non-finite** postmodifiers are the gerundive (*-ing*), *-ed*, and infinitive forms.

Gerundive

Gerundive postmodifiers are so-called because they consist of a verb phrase that begins with the gerundive (*-ing*) form of the verb. In the following examples, the verb phrases happen to all have only prepositional phrases after the verb, but in general this verb phrase can have anything in it (anything, that is, which is semantically and syntactically compatible with the gerund verb).

any of those *[leaving on Thursday]*
any flights *[arriving after eleven a.m.]*
flights *[arriving within thirty minutes of each other]*

We can define the *Nominals* with gerundive modifiers as follows, making use of a new non-terminal *GerundVP*:

$$Nominal \rightarrow Nominal\ GerundVP$$

We can make rules for *GerundVP* constituents by duplicating all of our VP productions, substituting *GerundV* for *V*.

$$\begin{aligned} GerundVP &\rightarrow GerundV\ NP \\ &\mid\ \ GerundV\ PP \mid GerundV \mid GerundV\ NP\ PP \end{aligned}$$

GerundV can then be defined as:

$$GerundV \rightarrow being \mid arriving \mid leaving \mid \ldots$$

The phrases in italics below are examples of the two other common kinds of non-finite clauses, infinitives and *-ed* forms:

the last flight *to arrive in Boston*
I need to have dinner *served*
Which is the aircraft *used by this flight*?

A postnominal relative clause (more correctly a **restrictive relative clause**), is a clause that often begins with a **relative pronoun** (*that* and *who* are the most common). The relative pronoun functions as the subject of the embedded verb (is a **subject relative**) in the following examples:

Relative pronoun

a flight *that serves breakfast*
flights *that leave in the morning*
the United flight *that arrives in San Jose around ten p.m.*
the one *that leaves at ten thirty five*

We might add rules like the following to deal with these:

$$\begin{aligned} Nominal &\rightarrow Nominal\ RelClause \\ RelClause &\rightarrow (who \mid that)\ VP \end{aligned}$$

The relative pronoun may also function as the object of the embedded verb, as in the following example; we leave as an exercise for the reader writing grammar rules for more complex relative clauses of this kind.

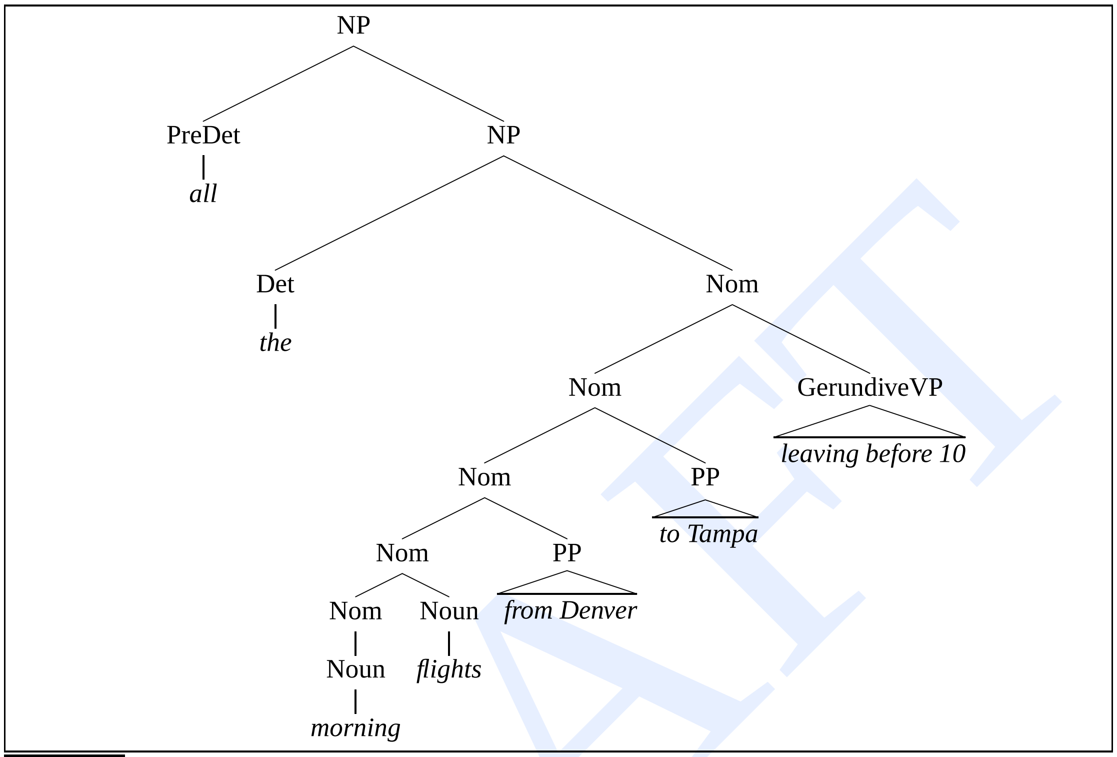


Figure 12.5 A parse tree for “all the morning flights from Denver to Tampa leaving before 10”.

the earliest American Airlines flight that I can get

Various postnominal modifiers can be combined, as the following examples show:

a flight *[from Phoenix to Detroit] [leaving Monday evening]*
I need a flight *[to Seattle] [leaving from Baltimore] [making a stop in Minneapolis]*
evening flights *[from Nashville to Houston] [that serve dinner]*
a friend *[living in Denver] [that would like to visit me here in Washington DC]*

Before the Noun Phrase

Predeterminers

Word classes that modify and appear before *NP*s are called **predeterminers**. Many of these have to do with number or amount; a common predeterminer is *all*:

all the flights all flights all non-stop flights

The example noun phrase given in Fig. 12.5 illustrates some of the complexity that arises when these rules are combined.

12.3.4 Agreement

In Ch. 3 we discussed English inflectional morphology. Recall that most verbs in English can appear in two forms in the present tense: the form used for third-person, singular subjects (*the flight does*), and the form used for all other kinds of subjects (*all* *the flights* <u>do</u>, *I* <u>do</u>). The third-person-singular (*3sg*) form usually has a final *-s* where the non-3sg form does not. Here are some examples, again using the verb *do*, with various subjects:

Do [$_{NP}$ all of these flights] offer first class service?
Do [$_{NP}$ I] get dinner on this flight?
Do [$_{NP}$ you] have a flight from Boston to Forth Worth?
Does [$_{NP}$ this flight] stop in Dallas?

Here are more examples with the verb *leave*:

What flights *leave* in the morning?
What flight *leaves* from Pittsburgh?

This agreement phenomenon occurs whenever there is a verb that has some noun acting as its subject. Note that sentences in which the subject does not agree with the verb are ungrammatical:

*[What flight] *leave* in the morning?
*Does [$_{NP}$ you] have a flight from Boston to Forth Worth?
*Do [$_{NP}$ this flight] stop in Dallas?

How can we modify our grammar to handle these agreement phenomena? One way is to expand our grammar with multiple sets of rules, one rule set for *3sg* subjects, and one for non-*3sg* subjects. For example, the rule that handled these yes-no-questions used to look like this:

$$S \rightarrow Aux\ NP\ VP$$

We could replace this with two rules of the following form:

$$\begin{aligned} S &\rightarrow 3sgAux\ 3sgNP\ VP \\ S &\rightarrow Non3sgAux\ Non3sgNP\ VP \end{aligned}$$

We could then add rules for the lexicon like these:

$$\begin{aligned} 3sgAux &\rightarrow does\ |\ has\ |\ can\ |\ \ldots \\ Non3sgAux &\rightarrow do\ |\ have\ |\ can\ |\ \ldots \end{aligned}$$

But we would also need to add rules for *3sgNP* and *Non3sgNP*, again by making two copies of each rule for *NP*. While pronouns can be first, second, or third person, full lexical noun phrases can only be third person, so for them we just need to distinguish between singular and plural (dealing with the first and second person pronouns is left as an exercise):

$$\begin{aligned} 3SgNP &\rightarrow Det\ SgNominal \\ Non3SgNP &\rightarrow Det\ PlNominal \\ SgNominal &\rightarrow SgNoun \\ PlNominal &\rightarrow PlNoun \\ SgNoun &\rightarrow flight\ |\ fare\ |\ dollar\ |\ reservation\ |\ \ldots \\ PlNoun &\rightarrow flights\ |\ fares\ |\ dollars\ |\ reservations\ |\ \ldots \end{aligned}$$

The problem with this method of dealing with number agreement is that it doubles the size of the grammar. Every rule that refers to a noun or a verb needs to have a "singular" version and a "plural" version. Unfortunately, subject-verb agreement is only the tip of the iceberg. We'll also have to introduce copies of rules to capture the fact that head nouns and their determiners have to agree in number as well:

this flight	*this flights
those flights	*those flight

Case

Nominative

Accusative

Rule proliferation will also have to happen for the noun's **case**; for example English pronouns have **nominative** (*I, she, he, they*) and **accusative** (*me, her, him, them*) versions. We will need new versions of every *NP* and *N* rule for each of these.

Gender agreement

These problems are compounded in languages like German or French, which not only have number-agreement as in English, but also have **gender agreement**. We mentioned briefly in Ch. 3 that the gender of a noun must agree with the gender of its modifying adjective and determiner. This adds another multiplier to the rule sets of the language.

Ch. 16 will introduce a way to deal with these agreement problems without exploding the size of the grammar, by effectively **parameterizing** each non-terminal of the grammar with **feature structures** and **unification**. But for many practical computational grammars, we simply rely on CFGs and make do with the large numbers of rules.

12.3.5 The Verb Phrase and Subcategorization

The verb phrase consists of the verb and a number of other constituents. In the simple rules we have built so far, these other constituents include *NP*s and *PP*s and combinations of the two:

$VP \rightarrow Verb$	disappear
$VP \rightarrow Verb\ NP$	prefer a morning flight
$VP \rightarrow Verb\ NP\ PP$	leave Boston in the morning
$VP \rightarrow Verb\ PP$	leaving on Thursday

Sentential complements

Verb phrases can be significantly more complicated than this. Many other kinds of constituents can follow the verb, such as an entire embedded sentence. These are called **sentential complements**:

You [$_{VP}$ [$_{V}$ said [$_{S}$ there were two flights that were the cheapest]]]
You [$_{VP}$ [$_{V}$ said [$_{S}$ you had a two hundred sixty six dollar fare]]
[$_{VP}$ [$_{V}$ Tell] [$_{NP}$ me] [$_{S}$ how to get from the airport in Philadelphia to downtown]]
I [$_{VP}$ [$_{V}$ think [$_{S}$ I would like to take the nine thirty flight]]

Here's a rule for these:

$$VP \rightarrow Verb\ S$$

Another potential constituent of the VP is another VP. This is often the case for verbs like *want, would like, try, intend, need*:

I want [$_{VP}$ to fly from Milwaukee to Orlando]
Hi, I want [$_{VP}$ to arrange three flights]
Hello, I'm trying [$_{VP}$ to find a flight that goes from Pittsburgh to Denver after two p.m.]

Recall from Ch. 5 that verbs can also be followed by *particles*, words that resemble a preposition but that combine with the verb to form a *phrasal verb* like *take off*. These particles are generally considered to be an integral part of the verb in a way that other post-verbal elements are not; phrasal verbs are treated as individual verbs composed of two words.

While a verb phrase can have many possible kinds of constituents, not every verb is compatible with every verb phrase. For example, the verb *want* can either be used with an NP complement (*I want a flight* ...), or with an infinitive VP complement (*I want to fly to* ...). By contrast, a verb like *find* cannot take this sort of VP complement. (* *I found to fly to Dallas*).

This idea that verbs are compatible with different kinds of complements is a very old one; traditional grammar distinguishes between **transitive** verbs like *find*, which take a direct object NP (*I found a flight*), and **intransitive** verbs like *disappear*, which do not (**I disappeared a flight*).

Transitive
Intransitive
Subcategorize

Where traditional grammars **subcategorize** verbs into these two categories (transitive and intransitive), modern grammars distinguish as many as 100 subcategories. (In fact, tagsets for many such subcategorization frames exist; see Macleod et al. (1998) for the COMLEX tagset, Sanfilippo (1993) for the ACQUILEX tagset, and further discussion in Ch. 16). We say that a verb like *find* **subcategorizes for** an *NP*, while a verb like *want* subcategorizes for either an *NP* or a non-finite *VP*. We also call these constituents the **complements** of the verb (hence our use of the term **sentential complement** above). So we say that *want* can take a *VP* complement. These possible sets of complements are called the **subcategorization frame** for the verb. Another way of talking about the relation between the verb and these other constituents is to think of the verb as a logical predicate and the constituents as logical arguments of the predicate. So we can think of such predicate-argument relations as FIND(I, A FLIGHT), or WANT(I, TO FLY). We will talk more about this view of verbs and arguments in Ch. 17 when we talk about predicate calculus representations of verb semantics.

Subcategorizes for
Complements
Subcategorization frame

Frame	**Verb**	**Example**
∅	eat, sleep	I ate
NP	prefer, find, leave	Find [$_{NP}$ the flight from Pittsburgh to Boston]
NP NP	show, give	Show [$_{NP}$ me] [$_{NP}$ airlines with flights from Pittsburgh]
PP_{from} PP_{to}	fly, travel	I would like to fly [$_{PP}$ from Boston] [$_{PP}$ to Philadelphia]
NP PP_{with}	help, load	Can you help [$_{NP}$ me] [$_{PP}$ with a flight]
VPto	prefer, want, need	I would prefer [$_{VPto}$ to go by United airlines]
VPbrst	can, would, might	I can [$_{VPbrst}$ go from Boston]
S	mean	Does this mean [$_{S}$ AA has a hub in Boston]?

Figure 12.6 Subcategorization frames for a set of example verbs.

Subcategorization frames for a set of example verbs are given in Fig. 12.6. Note that a verb can subcategorize for a particular type of verb phrase, such as a verb phrase whose verb is an infinitive (*VPto*), or a verb phrase whose verb is a bare stem (uninflected: *VPbrst*). Note also that a single verb can take different subcategorization frames. The verb *find*, for example, can take an *NP NP* frame (*find me a flight*) as well as an *NP* frame.

How can we represent the relation between verbs and their complements in a context-free grammar? One thing we could do is to do what we did with agreement features: make separate subtypes of the class Verb (*Verb-with-NP-complement*, *Verb-with-Inf-VP-complement*, *Verb-with-S-complement*, and so on):

$$\begin{aligned}
\textit{Verb-with-NP-complement} &\rightarrow \textit{find} \mid \textit{leave} \mid \textit{repeat} \mid \ldots \\
\textit{Verb-with-S-complement} &\rightarrow \textit{think} \mid \textit{believe} \mid \textit{say} \mid \ldots \\
\textit{Verb-with-Inf-VP-complement} &\rightarrow \textit{want} \mid \textit{try} \mid \textit{need} \mid \ldots
\end{aligned}$$

Then each *VP* rule could be modified to require the appropriate verb subtype:

$$\begin{aligned}
\textit{VP} &\rightarrow \textit{Verb-with-no-complement} && \text{disappear} \\
\textit{VP} &\rightarrow \textit{Verb-with-NP-comp NP} && \text{prefer a morning flight} \\
\textit{VP} &\rightarrow \textit{Verb-with-S-comp S} && \text{said there were two flights}
\end{aligned}$$

The problem with this approach, as with the same solution to the agreement feature problem, is a vast explosion in the number of rules. The standard solution to both of these problems is the **feature structure**, which will be introduced in Ch. 16 where we will also discuss the fact that nouns, adjectives, and prepositions can subcategorize for complements just as verbs can.

12.3.6 Auxiliaries

Auxiliaries

Modal verb

Perfect

Progressive

Passive

The subclass of verbs called **auxiliaries** or **helping verbs** have particular syntactic constraints which can be viewed as a kind of subcategorization. Auxiliaries include the **modal** verbs *can*, *could*, *may*, *might*, *must*, *will*, *would*, *shall*, and *should*, the **perfect** auxiliary *have*, the **progressive** auxiliary *be*, and the **passive** auxiliary *be*. Each of these verbs places a constraint on the form of the following verb, and each of these must also combine in a particular order.

Modal verbs subcategorize for a *VP* whose head verb is a bare stem; for example, *can go in the morning*, *will try to find a flight*. The perfect verb *have* subcategorizes for a *VP* whose head verb is the past participle form: *have booked 3 flights*. The progressive verb *be* subcategorizes for a *VP* whose head verb is the gerundive participle: *am going from Atlanta*. The passive verb *be* subcategorizes for a *VP* whose head verb is the past participle: *was delayed by inclement weather*.

A sentence can have multiple auxiliary verbs, but they must occur in a particular order: *modal* $<$ *perfect* $<$ *progressive* $<$ *passive*. Here are some examples of multiple auxiliaries:

modal perfect	*could have been* a contender
modal passive	*will be* married
perfect progressive	*have been* feasting
modal perfect passive	*might have been* prevented

Auxiliaries are often treated just like verbs such as *want*, *seem*, or *intend*, which subcategorize for particular kinds of *VP* complements. Thus *can* would be listed in the lexicon as a *verb-with-bare-stem-VP-complement*. One way of capturing the ordering constraints among auxiliaries, commonly used in the **systemic grammar** of Halliday (1985), is to introduce a special constituent called the **verb group**, whose subconstituents include all the auxiliaries as well as the main verb. Some of the ordering constraints can also be captured in a different way. Since modals, for example, do not have a progressive or participle form, they simply will never be allowed to follow progressive or passive *be* or perfect *have*. Exercise 8 asks the reader to write grammar rules for auxiliaries.

Systemic grammar

Verb group

The passive construction has a number of properties that make it different than other auxiliaries. One important difference is a semantic one; while the subject of non-passive (**active**) sentence is often the semantic agent of the event described by the verb ($\underline{I}$ *prevented a catastrophe*) the subject of the passive is often the undergoer or patient of the event ($\underline{\textit{a catastrophe}}$ *was prevented*). This will be discussed further in Ch. 18.

Active

12.3.7 Coordination

Conjunctions

Coordinate

The major phrase types discussed here can be conjoined with **conjunctions** like *and*, *or*, and *but* to form larger constructions of the same type. For example a **coordinate** noun phrase can consist of two other noun phrases separated by a conjunction:

Please repeat [$_{NP}$ [$_{NP}$ the flights] *and* [$_{NP}$ the costs]]
I need to know [$_{NP}$ [$_{NP}$ the aircraft] *and* [$_{NP}$ the flight number]]

Here's a rule that allows these structures:

$$NP \rightarrow NP \textit{ and } NP$$

Note that the ability to form coordinate phrases via conjunctions is often used as a test for constituency. Consider the following examples which differ from the ones given above in that they lack the second determiner.

Please repeat the [$_{Nom}$ [$_{Nom}$ flights] *and* [$_{Nom}$ costs]]
I need to know the [$_{Nom}$ [$_{Nom}$ aircraft] *and* [$_{Nom}$ flight number]]

The fact that these phrases can be conjoined is evidence for the presence of the underlying *Nominal* constituent we have been making use of. Here's a new rule for this:

$$Nominal \rightarrow Nominal \textit{ and } Nominal$$

The following examples illustrate conjunctions involving *VP*s and *S*s.

What flights do you have [$_{VP}$ [$_{VP}$ leaving Denver] *and* [$_{VP}$ arriving in San Francisco]]
[$_{S}$ [$_{S}$ I'm interested in a flight from Dallas to Washington] *and* [$_{S}$ I'm also interested in going to Baltimore]]

The rules for *VP* and *S* conjunctions mirror the *NP* one given above.

$$VP \rightarrow VP \textit{ and } VP$$
$$S \rightarrow S \textit{ and } S$$

Since all the major phrase types can be conjoined in this fashion it is also possible to represent this conjunction fact more generally; a number of grammar formalisms such as (Gazdar et al., 1985) do this via **metarules** such as the following:

Metarules

$$X \rightarrow X \text{ and } X$$

This metarule simply states that any non-terminal can be conjoined with the same non-terminal to yield a constituent of the same type. Of course, the variable X must be designated as a variable that stands for any non-terminal rather than a non-terminal itself.

12.4 Treebanks

Context-free grammar rules of the type that we have explored so far in this chapter can be used, in principle, to assign a parse tree to any sentence. This means that it is possible to build a corpus in which every sentence is syntactically annotated with a parse tree. Such a syntactically annotated corpus is called a **treebank**. Treebanks play an important roles in parsing, as we will see in Ch. 13, and in various empirical investigations of syntactic phenomena.

Treebank

A wide variety of treebanks have been created, generally by using parsers (of the sort described in the next two chapters) to automatically parse each sentence, and then using humans (linguists) to hand-correct the parses. The **Penn Treebank** project (whose POS tagset we introduced in Ch. 5) has produced treebanks from the Brown, Switchboard, ATIS, and Wall Street Journal corpora of English, as well as treebanks in Arabic and Chinese. Other treebanks include the Prague Dependency Treebank for Czech, the Negra treebank for German, and the Susanne treebank for English.

Penn Treebank

12.4.1 Example: The Penn Treebank Project

Fig. 12.7 shows sentences from the Brown and ATIS portions of the Penn Treebank.[1] Note the formatting differences for the part-of-speech tags; such small differences are common and must be dealt with in processing treebanks. The Penn Treebank part-of-speech tagset was defined in Ch. 5. The use of LISP-style parenthesized notation for trees is extremely common, and resembles the bracketed notation we saw above in (12.2). For those who are not familiar with it we show a standard node-and-line tree representation in Fig. 12.8.

Fig. 12.9 shows a tree from the Wall Street Journal. This tree shows another feature of the Penn Treebanks: the use of **traces** (`-NONE-` nodes) to mark long-distance dependencies or **syntactic movement**. For example, quotations often follow a quotative verb like *say*. But in this example the quotation "We would have to wait until we have collected on those assets" precedes the words *he said*. An empty S containing only

Traces

Syntactic movement

[1] The Penn Treebank project released treebanks in multiple languages and in various stages; for example there were Treebank I (Marcus et al., 1993), Treebank II (Marcus et al., 1994), and Treebank III releases of English treebanks. We will use Treebank III for our examples.

```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```
(a)

```
((S
   (NP-SBJ The/DT flight/NN )
   (VP should/MD
     (VP arrive/VB
       (PP-TMP at/IN
         (NP eleven/CD a.m/RB ))
       (NP-TMP tomorrow/NN )))))
```
(b)

Figure 12.7 Parsed sentences from the LDC Treebank3 version of the Brown (a) and ATIS (b) corpora.

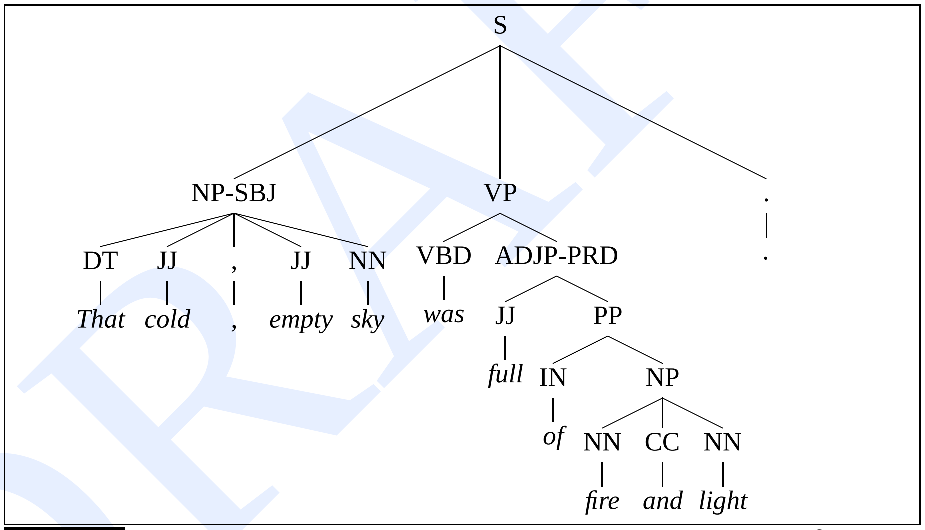


Figure 12.8 The tree corresponding to the Brown corpus sentence in the previous figure.

the node `-NONE-` is used to mark the position after *said* where the quotation sentence often occurs. This empty node is marked (in Treebanks II and III) with the index 2, as is the quotation S at the beginning of the sentence. Such coindexing may make it easier for some parsers to recover the fact that this fronted or topicalized quotation is the complement of the verb *said*. A similar `-NONE-` node is used mark the fact that there is no syntactic subject right before the verb *to wait*; instead, the subject is the earlier NP *We*. Again, they are both coindexed with the index 1.

The Penn Treebank II and Treebank III releases added further information to make it easier to recover the relationships between predicates and arguments. Certain phrases were marked with tags indicating the grammatical function of the phrase (as surface subject, logical topic, cleft, non-VP predicates) whether it appeared in particular text categories (headlines, titles), and its semantic function (temporal phrases, locations)

```
( (S (‘‘ ‘‘)
    (S-TPC-2
      (NP-SBJ-1 (PRP We) )
      (VP (MD would)
        (VP (VB have)
          (S
            (NP-SBJ (-NONE- *-1) )
            (VP (TO to)
              (VP (VB wait)
                (SBAR-TMP (IN until)
                  (S
                    (NP-SBJ (PRP we) )
                    (VP (VBP have)
                      (VP (VBN collected)
                        (PP-CLR (IN on)
                          (NP (DT those)(NNS assets)))))))))))))
    (, ,) (’’ ’’)
    (NP-SBJ (PRP he) )
    (VP (VBD said)
      (S (-NONE- *T*-2) ))
    (. .) ))
```

Figure 12.9 A sentence from the Wall Street Journal portion of the LDC Penn Treebank. Note the use of the empty `-NONE-` nodes.

(Marcus et al., 1994; Bies et al., 1995). Fig. 12.9 shows examples of the `-SBJ` (surface subject) and `-TMP` (temporal phrase) tags. Fig. 12.8 shows in addition the `-PRD` tag, which is used for predicates which are not VPs (the one in Fig. 12.8 is an ADJP). Fig. 12.20 shows the tag `-UNF` in `NP-UNF` meaning ‘unfinished or incomplete phrase’.

12.4.2 Using a Treebank as a Grammar

The sentences in a treebank implicitly constitute a grammar of the language. For example, we can take the three parsed sentences in Fig. 12.7 and Fig. 12.9 and extract each of the CFG rules in them. For simplicity, let’s strip off the rule suffixes (`-SBJ` and so on). The resulting grammar is shown in Fig. 12.10.

The grammar used to parse the Penn Treebank is relatively flat, resulting in very many and very long rules. For example among the approximately 4,500 different rules for expanding VP are separate rules for PP sequences of any length, and every possible arrangement of verb arguments:

```
VP → VBD PP
VP → VBD PP PP
VP → VBD PP PP PP
VP → VBD PP PP PP PP
VP → VB ADVP PP
VP → VB PP ADVP
```

Grammar	Lexicon
S → *NP VP .*	*PRP* → *we* | *he*
S → *NP VP*	*DT* → *the* | *that* | *those*
S → *" S " , NP VP .*	*JJ* → *cold* | *empty* | *full*
S → *-NONE-*	*NN* → *sky* | *fire* | *light* | *flight* | *tomorrow*
NP → *DT NN*	*NNS* → *assets*
NP → *DT NNS*	*CC* → *and*
NP → *NN CC NN*	*IN* → *of* | *at* | *until* | *on*
NP → *CD RB*	*CD* → *eleven*
NP → *DT JJ , JJ NN*	*RB* → *a.m.*
NP → *PRP*	*VB* → *arrive* | *have* | *wait*
NP → *-NONE-*	*VBD* → *was* | *said*
VP → *MD VP*	*VBP* → *have*
VP → *VBD ADJP*	*VBN* → *collected*
VP → *VBD S*	*MD* → *should* | *would*
VP → *VBN PP*	*TO* → *to*
VP → *VB S*	
VP → *VB SBAR*	
VP → *VBP VP*	
VP → *VBN PP*	
VP → *TO VP*	
SBAR → *IN S*	
ADJP → *JJ PP*	
PP → *IN NP*	

Figure 12.10 A sample of the CFG grammar rules and lexical entries that would be extracted from the three treebank sentences in Fig. 12.7 and Fig. 12.9.

```
VP → ADVP VB PP
```

as well as even longer rules, such a:

```
VP → VBP PP PP PP PP PP ADVP PP
```

which comes from the VP marked in italics:

(12.7) This mostly happens because we *go from football in the fall to lifting in the winter to football again in the spring*.

Some of the many thousands of NP rules include:

```
NP → DT JJ NN
NP → DT JJ NNS
NP → DT JJ NN NN
NP → DT JJ JJ NN
NP → DT JJ CD NNS
NP → RB DT JJ NN NN
NP → RB DT JJ JJ NNS
NP → DT JJ JJ NNP NNS
NP → DT NNP NNP NNP NNP JJ NN
NP → DT JJ NNP CC JJ JJ NN NNS
NP → RB DT JJS NN NN SBAR
NP → DT VBG JJ NNP NNP CC NNP
NP → DT JJ NNS , NNS CC NN NNS NN
NP → DT JJ JJ VBG NN NNP NNP FW NNP
NP → NP JJ , JJ ‘‘ SBAR ’’ NNS
```

The last two of those rules, for example, come from the following two NPs:

(12.8) [$_{DT}$ The] [$_{JJ}$ state-owned] [$_{JJ}$ industrial] [$_{VBG}$ holding] [$_{NN}$ company] [$_{NNP}$ Instituto] [$_{NNP}$ Nacional] [$_{FW}$ de] [$_{NNP}$ Industria]

(12.9) [$_{NP}$ Shearson’s] [$_{JJ}$ easy-to-film], [$_{JJ}$ black-and-white] “[$_{SBAR}$ Where We Stand]” [$_{NNS}$ commercials]

Viewed as a large grammar in this way, the Penn Treebank III Wall Street Journal corpus, which contains about 1 million words, also has about 1 million non-lexical rule tokens, consisting of about 17,500 distinct rule types.

Various facts about the treebank grammars, such as their large numbers of flat rules, pose problems for probabilistic parsing algorithms. For this reason, it is common to make various modifications to a grammar extracted from a treebank. We will discuss these further in Ch. 14.

12.4.3 Searching Treebanks

It is often important to search through a treebank to find examples of particular grammatical phenomena, either for linguistic research or for answering analytic questions about a computational application. But neither the regular expressions used for text search nor the boolean expressions over words used for web search are a sufficient search tool. What is needed is a language that can specify constraints about nodes and links in a parse tree, so as to search for specific patterns.

Various such tree-searching languages exist in different tools. **Tgrep** (Pito, 1993) and **TGrep2** (Rohde, 2005) are publicly-available tools for searching treebanks that use a similar language for expressing tree constraints. We’ll describe the more recent language used by **TGrep2**, drawing from the online manual (Rohde, 2005).

A pattern in **tgrep** or **TGrep2** consists of a specification of a node, possibly followed by links to other nodes. A node specification can then be used to return the subtree rooted at that node. For example, the pattern

```
NP
```

returns all subtrees in a corpus whose root is NP. Nodes can be specified by a name, a regular expression inside slashes, or a disjunction of these. For example, we can specify a singular or plural noun (`NN` or `NNS`) using Penn Treebank notation as either of the following:

```
/NNS?/        NN|NNS
```

A node which either is the word *bush* or else ends in the string *tree* can be expressed as:

```
/tree$/|bush
```

The power of **tgrep/TGrep2** patterns is the ability to specify information about links. The operator `<` means **immediately dominates**; the following pattern thus matches an NP immediately dominating a PP

```
NP < PP
```

The relation << is used to specify dominance; this pattern matches an NP dominating a PP:

```
NP << PP
```

This previous pattern would thus match either of the following trees:

(12.10)
```
(NP (NP (NN reinvestment))
    (PP (IN of)
        (NP (NNS dividends))))
```

(12.11)
```
(NP (NP (DT the) (JJ austere) (NN company) (NN dormitory))
    (VP (VBN run)
        (PP (IN by) (NP (DT a) (JJ prying) (NN caretaker)))))
```

The relation `.` is used to mark linear precedence. The following pattern matches an NP that immediately dominates a JJ and is immediately followed by a PP, for example matching the NP dominating *the austere company dormitory* in (12.11) above:[2]

```
NP < JJ . VP
```

Each of the relations in a **tgrep/TGrep2** expression is interpreted as referring to the first or root node. Thus for example the following expression means an NP which both precedes a PP and dominates an S:

```
NP . PP < S
```

If we wanted instead to specify that the PP dominated the S, we could use parentheses as follows:

```
NP . (PP < S)
```

Fig. 12.11 gives the major link operations for **TGrep2**.

12.4.4 Heads and Head Finding

We suggested informally earlier that syntactic constituents could be associated with a lexical **head**; *N* is the head of an *NP*, *V* is the head of a *VP*. This idea of a head for each constituent dates back to Bloomfield (1914). It is central to such linguistic formalisms such as Head-Driven Phrase Structure Grammar (Pollard and Sag, 1994), and has become extremely popular in computational linguistics with the rise of lexicalized grammars (Ch. 14).

In one simple model of lexical heads, each context-free rule is associated with a head (Charniak, 1997; Collins, 1999). The head is the word in the phrase which is grammatically the most important. Heads are passed up the parse tree; thus each non-terminal in a parse-tree is annotated with a single word which is its lexical head. Fig. 12.12 shows an example of such a tree from Collins (1999), in which each non-terminal is annotated with its head. “Workers dumped sacks into a bin” is a shortened form of a WSJ sentence.

In order to generate such a tree, each CFG rule must be augmented to identify one right-hand-side constituent to be the head daughter. The headword for a node is then

[2] The definition of linear precedence differs slightly between **tgrep** and **TGrep2**. See Rohde (2005) for more details.

Link	Explanation
`A < B`	A is the parent of (immediately dominates) B.
`A > B`	A is the child of B.
`A <N B`	B is the Nth child of A (the first child is `<1`).
`A >N B`	A is the Nth child of B (the first child is `>1`).
`A <, B`	Synonymous with `A <1 B`.
`A >, B`	Synonymous with `A >1 B`.
`A <-N B`	B is the Nth-to-last child of A (the last child is `<-1`).
`A >-N B`	A is the Nth-to-last child of B (the last child is `>-1`).
`A <- B`	B is the last child of A (synonymous with `A <-1 B`).
`A >- B`	A is the last child of B (synonymous with `A >-1 B`).
`A <‘ B`	B is the last child of A (also synonymous with `A <-1 B`).
`A >‘ B`	A is the last child of B (also synonymous with `A >-1 B`).
`A <: B`	B is the only child of A
`A >: B`	A is the only child of B
`A << B`	A dominates B (A is an ancestor of B).
`A >> B`	A is dominated by B (A is a descendant of B).
`A <<, B`	B is a left-most descendant of A.
`A >>, B`	A is a left-most descendant of B.
`A <<‘ B`	B is a right-most descendant of A.
`A >>‘ B`	A is a right-most descendant of B.
`A <<: B`	There is a single path of descent from A and B is on it.
`A >>: B`	There is a single path of descent from B and A is on it.
`A . B`	A immediately precedes B.
`A , B`	A immediately follows B.
`A .. B`	A precedes B.
`A ,, B`	A follows B.
`A $ B`	A is a sister of B (and A $\neq$ B).
`A $. B`	A is a sister of and immediately precedes B.
`A $, B`	A is a sister of and immediately follows B.
`A $.. B`	A is a sister of and precedes B.
`A $,, B`	A is a sister of and follows B.

Figure 12.11 Links in **TGrep2**, summarized from Rohde (2005).

set to the headword of its head daughter. Choosing these head daughters is simple for textbook examples (*NN* is the head of *NP*) but is complicated and indeed controversial for most phrases. (Should the complementizer *to* or the verb be the head of an infinite verb-phrase?) Modern linguistic theories of syntax generally include a component that defines heads (see e.g., Pollard and Sag, 1994).

An alternative approach to head-finding is used in most practical computational systems. Instead of specifying head rules in the grammar itself, heads are identified dynamically in the context of trees for specific sentences. In other words, once a sentence is parsed, the resulting tree is walked to decorate each node with the appropriate head. Most current systems rely on a simple set of hand-written rules, such as a practical one for Penn Treebank grammars given in Collins (1999) but developed originally by Magerman (1995b). For example their rule for finding the head of an NP is as follows Collins (1999, 238):

- If the last word is tagged POS, return last-word.

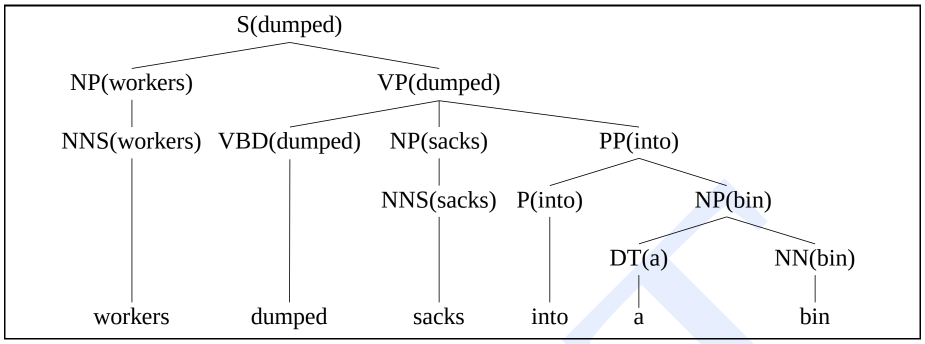


Figure 12.12 A lexicalized tree from Collins (1999).

- Else search from right to left for the first child which is an NN, NNP, NNPS, NX, POS, or JJR.
- Else search from left to right for the first child which is an NP.
- Else search from right to left for the first child which is a $, ADJP, or PRN.
- Else search from right to left for the first child which is a CD.
- Else search from right to left for the first child which is a JJ, JJS, RB or QP.
- Else return the last word

Selected other rules from their set are shown in Fig. 12.13. For example, for VP rules of the form $VP \rightarrow Y_1 \cdots Y_n$, the algorithm would start from the left of $Y_1 \cdots Y_n$ looking for the first Y_i of type TO; if no TOs are found it would search for the first Y_i of type VBD; if no VBDs are found it would search for a VBP, and so on. See Collins (1999) for more details.

Parent	**Direction**	**Priority List**
ADJP	Left	NNS QP NN $ ADVP JJ VBN VBG ADJP JJR NP JJS DT FW RBR RBS SBAR RB
ADVP	Right	RB RBR RBS FW ADVP TO CD JJR JJ IN NP JJS NN
PRN	Left	
PRT	Right	RP
QP	Left	$ IN NNS NN JJ RB DT CD NCD QP JJR JJS
S	Left	TO IN VP S SBAR ADJP UCP NP
SBAR	Left	WHNP WHPP WHADVP WHADJP IN DT S SQ SINV SBAR FRAG
VP	Left	TO VBD VBN MD VBZ VB VBG VBP VP ADJP NN NNS NP

Figure 12.13 Selected head rules from Collins (1999). The set of head rules is often called a **head percolation table**.

12.5 Grammar Equivalence and Normal Form

A formal language is defined as a (possibly infinite) set of strings of words. This suggests that we could ask if two grammars are equivalent by asking if they generate the same set of strings. In fact it is possible to have two distinct context-free grammars generate the same language.

We usually distinguish two kinds of grammar equivalence: **weak equivalence** and **strong equivalence**. Two grammars are strongly equivalent if they generate the same set of strings *and* if they assign the same phrase structure to each sentence (allowing merely for renaming of the non-terminal symbols). Two grammars are weakly equivalent if they generate the same set of strings but do not assign the same phrase structure to each sentence.

Normal form

It is sometimes useful to have a **normal form** for grammars, in which each of the productions takes a particular form. For example a context-free grammar is in **Chomsky Normal Form** (CNF) (Chomsky, 1963) if it is ϵ-free and if in addition each production is either of the form $A \rightarrow B\ C$ or $A \rightarrow a$. That is, the right-hand side of each rule either has two non-terminal symbols or one terminal symbol. Chomsky normal form grammars are **binary branching**, i.e. have binary trees (down to the prelexical nodes). We will make use of this binary branching property in the CKY parsing algorithm in Ch. 13.

Chomsky Normal Form

Binary branching

Any grammar can be converted into a weakly-equivalent Chomsky normal form grammar. For example, a rule of the form

$$A \rightarrow B\ C\ D$$

can be converted into the following two CNF rules (Exercise 11 asks the reader to formulate the complete algorithm):

$$\begin{aligned} A &\rightarrow B\ X \\ X &\rightarrow C\ D \end{aligned}$$

Sometimes using binary branching can actually produce smaller grammars. For example the sentences that might be characterized as follows:

```
VP -> VBD NP PP*
```

are represented in the Penn Treebank by this series of rules:

```
VP → VBD NP PP
VP → VBD NP PP PP
VP → VBD NP PP PP PP
VP → VBD NP PP PP PP PP
...
```

but could also be generated by the following two-rule grammar:

```
VP → VBD NP PP
VP → VP PP
```

Chomsky-adjunction

To generate a symbol A with a potentially infinite sequence of symbols B by using a rule of the form A → A B is known as **Chomsky-adjunction**.

12.6 Finite-State and Context-Free Grammars

We argued in Sec. 12.1 that adequate models of grammar need to be able to represent complex interrelated facts about constituency, subcategorization, and dependency relations, and we implied that at the least the power of context-free grammars is needed to accomplish this. But why is it that we can't just use finite-state methods to capture these syntactic facts? The answer to this question is critical since, as we'll see in Ch. 13, there is a considerable price to be paid in terms of processing speed when one switches from regular languages to context-free ones.

There are two answers to this question. The first is mathematical; we'll show in Ch. 15 that given certain assumptions, that certain syntactic structures present in English (and other natural languages) make them not regular languages. The second answer is more subjective and has to do with notions of expressiveness; even when finite-state methods are capable of dealing with the syntactic facts in question, they often don't express them in ways that make generalizations obvious, lead to understandable formalisms, or produce structures of immediate use in subsequent semantic processing.

The mathematical objection will be discussed more fully in Ch. 15, but we'll briefly review it here. We mentioned in passing in Ch. 2 that there is a completely equivalent alternative to finite-state machines and regular expressions for describing regular languages, called **regular grammars**. The rules in a regular grammar are a restricted form of the rules in a context-free grammar because they are in right-linear or left-linear form. In a right-linear grammar, for example, the rules are all of the form $A \rightarrow w*$ or $A \rightarrow w * B$, that is the non-terminals either expand to a string of terminals or to a string of terminals followed by a non-terminal. These rules look an awful lot like the rules we've been using throughout this chapter, so what can't they do? What they can't do is express recursive **center-embedding** rules like the following, where a non-terminal is rewritten as itself, surrounded by (non-empty) strings:

(12.12) $$A \overset{*}{\Rightarrow} \alpha A \beta$$

In other words, a language can be generated by a finite-state machine if and only if the grammar that generates L that does not have any **center-embedded** recursions of this form (Chomsky, 1959a; Bar-Hillel et al., 1961; Nederhof, 2000). Intuitively, this is because grammar rules in which the non-terminal symbols are always on either the right or left edge of a rule can be processed iteratively rather than recursively. Such center-embedding rules are needed to deal with artificial problems such as the language $a^n b^n$, or for practical problems such as checking for correctly matching delimiters in programming and markup languages. It turns out that there are no slam-dunk examples of this for English, but examples like the following give a flavor of the problem.

(12.13) The luggage arrived.

(12.14) The luggage that the passengers checked arrived.

(12.15) The luggage that the passengers that the storm delayed checked arrived.

At least in theory, this kind of embedding could go on, although it gets increasingly difficult to process such examples and they are luckily fairly rare outside textbooks like this one. Ch. 15 will discuss this and related issues as to whether or not even context-free grammars are up to the task.

So is there no role for finite-state methods in syntactic analysis? A quick review of the rules used for noun-phrases in this chapter, as well as those used in the Penn treebank grammar, reveals that a considerable portion of them can be handled by finite-state methods. Consider the following rule for a **noun group**, the pre-nominal and nominal portions of a noun phrase:

Noun group

$$Nominal \rightarrow (Det)\ (Card)\ (Ord)\ (Quant)\ (AP)\ Nominal$$

Assuming we convert the pre-nominal elements of this rule into terminals, this rule is effectively right-linear and can be captured by a finite-state machine. Indeed, it is possible to automatically build a regular grammar which is an approximation of a given context-free grammar; see the references at the end of the chapter. Thus for many practical purposes where matching syntactic and semantic rules aren't necessary, finite-state rules are quite sufficient.

12.7 Dependency Grammars

We have focused in this chapter on context-free grammars because many available treebanks and parsers produce these kinds of syntactic representation. But in a class of grammar formalisms called **dependency grammars** that are becoming quite important in speech and language processing, constituents and phrase-structure rules do not play any fundamental role. Instead, the syntactic structure of a sentence is described purely in terms of words and binary semantic or syntactic relations between these words. Dependency grammars often draw heavily from the work of Tesnière (1959), and the name **dependency** might have been used first by early computational linguist David Hays. But this lexical dependency notion of grammar is in fact older than the relatively recent phrase-structure or constituency grammars, and has its roots in the ancient Greek and Indian linguistic traditions. Indeed the notion in traditional grammar of "parsing a sentence into subject and predicate" is based on lexical relations rather than constituent relations.

Dependency grammar

Dependency

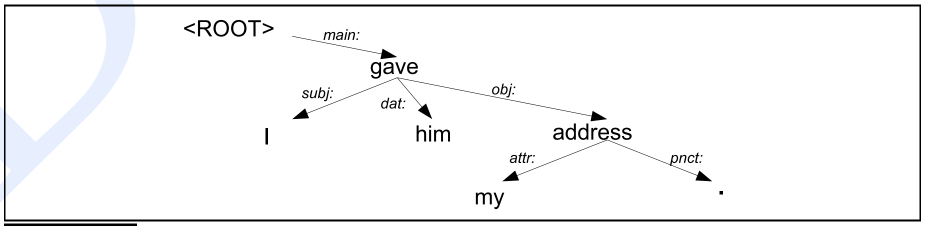


Figure 12.14 A sample dependency grammar parse, using the dependency formalism of Karlsson et al. (1995b), after Järvinen and Tapanainen (1997).

Fig. 12.14 shows an example parse of the sentence *I gave him my address*, using the dependency grammar formalism of Järvinen and Tapanainen (1997) and Karlsson et al. (1995b). Note that there are no non-terminal or phrasal nodes; each link in

Dependency	Description
subj	syntactic subject
obj	direct object (incl. sentential complements)
dat	indirect object
pcomp	complement of a preposition
comp	predicate nominals (complements of copulas)
tmp	temporal adverbials
loc	location adverbials
attr	premodifying (attributive) nominals (genitives, etc.)
mod	nominal postmodifiers (prepositional phrases, etc.)

Figure 12.15 A representative set of grammatical relations from (Järvinen and Tapanainen, 1997).

the parse tree holds between two lexical nodes (augmented with the special <ROOT> node). The links are drawn from a fixed inventory of around 35 relations, most of which roughly represent grammatical functions or very general semantic relations. Other dependency-based computational grammars, such as **Link Grammar** (Sleator and Temperley, 1993), use different but roughly overlapping links. Table 12.15 shows a few of the relations used in Järvinen and Tapanainen (1997):

Link Grammar

As we will see in Ch. 14, one advantage of dependency formalisms is the strong predictive parsing power that words have for their dependents. Knowing the identity of the verb is often a very useful cue for deciding which noun is likely to be the subject or the object. Dependency grammar researchers argue that one of the main advantages of pure dependency grammars is their ability to handle languages with relatively **free word order**. For example the word order in languages like Czech is much more flexible than in English; an *object* might occur before or after a *location adverbial* or a **comp**. A phrase-structure grammar would need a separate rule for each possible place in the parse tree that such an adverbial phrase could occur. A dependency grammar would just have one link-type representing this particular adverbial relation. Thus a dependency grammar abstracts away from word-order variation, representing only the information that is necessary for the parse.

Free word order

There are a number of computational implementations of dependency grammars; Link Grammar (Sleator and Temperley, 1993) and Constraint Grammar (Karlsson et al., 1995b) are easily-available broad-coverage dependency grammars and parsers for English. Dependency grammars are also often used for other languages. Hajič (1998), for example, describes the 500,000 word Prague Dependency Treebank for Czech which has been used to train probabilistic dependency parsers (Collins et al., 1999).

12.7.1 The Relationship Between Dependencies and Heads

The reader may have noticed the similarity between dependency graphs like Fig. 12.14 and head structures like Fig. 12.12. In fact an (unlabeled) dependency graph can be automatically derived from a context-free parse by using the head rules; here's an algorithm from Xia and Palmer (2001):

1. Mark the head child of each node in a phrase structure, using the head percolation table.

2. In the dependency structure, make the head of each non- head-child depend on the head of the head-child.

This algorithm applied to the parse tree in Fig. 12.16 would produce the dependency structure in Fig. 12.17.

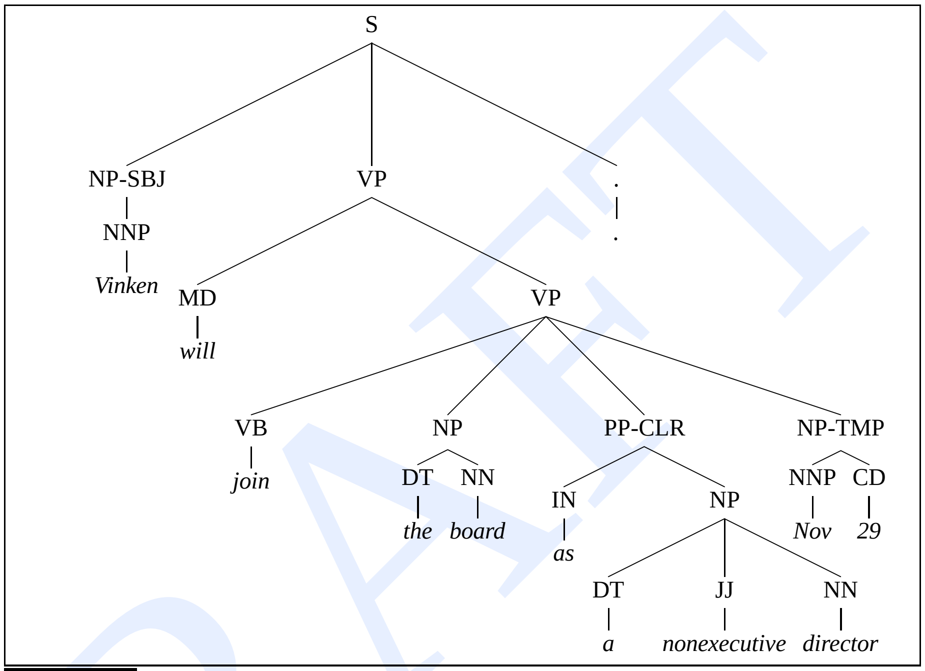


Figure 12.16 A phrase structure tree from the Wall Street Journal component of the Penn Treebank 3

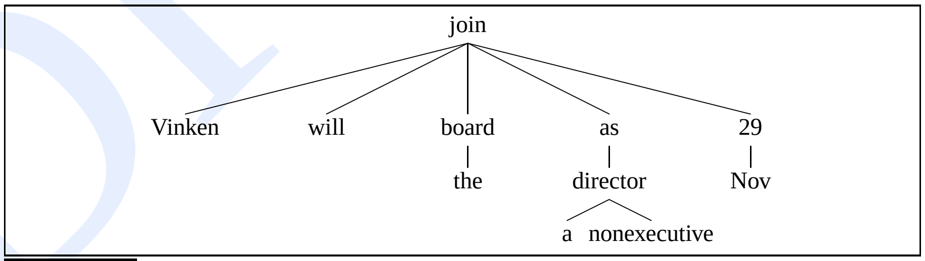


Figure 12.17 The dependency tree produced from Fig. 12.16 by the algorithm given above.

We will return to the discussion of heads and dependencies when we discuss lexicalized parsing in Ch. 14 and again when we introduce head features and subcategorization in Ch. 16.

12.7.2 Categorial Grammar

Categorial grammar

Categorial grammar is an early lexicalized grammar model (Adjukiewicz, 1935; Bar-

Combinatory categorial grammar

Hillel, 1953). In this section we will give a simplified overview of one important extension to categorial grammar, **combinatory categorial grammar**, or CCG (Steedman, 1996, 1989, 2000). A categorial grammar has two components. The **categorial lexicon** associates each word with a syntactic and semantic category. The **combinatory rules** allow functions and arguments to be combined. There are two types of categories: functors and arguments. Arguments, like nouns, have simple categories like N. Verbs or determiners act as functors. For example, a determiner can be thought of as a function that applies to an N on its right to produce an NP. Such complex categories are built using the X/Y and X\Y operators. X/Y means a function from Y to X, that is, something which combines with a Y on its right to produce an X. Determiners thus receive the category NP/N: something that combines with an N on its right to produce an NP. Transitive verbs might have the category VP/NP; something that combines with an NP on the right to produce a VP. Ditransitive verbs like *give* might have the category (VP/NP)/NP; something which combines with an NP on its right to yield a transitive verb. The simplest **combination rules** just combine an X/Y with a Y on its right to produce an X or a X\Y with a Y on its left to produce an X.

Consider the simple sentence *Harry eats apples* from Steedman (1989). Instead of using a primitive VP category, let's assume that a finite verb phrase like *eat apples* has the category (S\NP); something which combines with an NP on the left to produce a sentence. *Harry* and *apples* are both NPs. *Eats* is a finite transitive verb which combines with an NP on the right to produce a finite VP: (S\NP)/NP. The derivation of S proceeds as follows:

(12.16)
$$\begin{array}{lll}
\underline{\text{Harry}} & \underline{\text{eats}} & \underline{\text{apples}} \\
\text{NP} & (\text{S}\backslash\text{NP})/\text{NP} & \text{NP} \\
\cline{2-3}
 & \multicolumn{2}{c}{\text{S}\backslash\text{NP}} \\
\cline{1-2}
\multicolumn{2}{c}{\text{S}} &
\end{array}$$

Modern categorial grammars include more complex combinatory rules which are needed for coordination and other complex phenomena, and also include composition of semantic categories as well as syntactic ones. See the end of the chapter for a pointer to useful references.

12.8 Spoken Language Syntax

The grammar of written English and the grammar of conversational spoken English share many features, but also differ in a number of respects. This section gives a quick sketch of a number of the characteristics of the syntax of spoken English.

Utterance

We usually use the term **utterance** rather than **sentence** for the units of spoken language. Fig. 12.18 shows some sample spoken ATIS utterances that exhibit many aspects of spoken language grammar.

This is a standard style of transcription used in transcribing speech corpora for speech recognition. The comma "," marks a short pause, and each period "." marks a long pause. **Fragments** (incomplete words like *wha-* for incomplete *what*) are marked with a dash, and the square brackets "[smack]" mark non-verbal events (**lipsmacks**, the . [exhale] . . . [inhale] . . uh does American airlines . offer any . one way flights . uh one way fares, for one hundred and sixty one dollars

[mm] i'd like to leave i guess between um . [smack] . five o'clock no, five o'clock and uh, seven o'clock . P M

all right, [throat_clear] . . i'd like to know the . give me the flight . times . in the morning . for September twentieth . nineteen ninety one

uh one way

. w- wha- what is the lowest, cost, fare

[click] . i need to fly, betwee- . leaving . Philadelphia . to, Atlanta [exhale]

on United airlines . . give me, the . . time . . from New York . [smack] . to Boise-, to . I'm sorry . on United airlines . [uh] give me the flight, numbers, the flight times from . [uh] Boston . to Dallas

Figure 12.18 Sample spoken utterances from users interacting with an ATIS system.

breaths, etc.).

There are a number of ways these utterances differ from written English sentences. One is in the lexical statistics; for example spoken English is much higher in pronouns than written English; the subject of a spoken sentence is almost invariably a pronoun. Spoken sentences often consist of short fragments or phrases (*one way* or *around four p.m.*, which are less common in written English. Spoken sentences have phonological, prosodic, and acoustic characteristics that of course written utterances don't have; we will return to these in Ch. 8. Finally, spoken sentences have various kinds of disfluencies (hesitations, repairs, restarts, etc) to be discussed below.

12.8.1 Disfluencies and Repair

Perhaps the most salient syntactic feature that distinguishes spoken and written language is the class of phenomena known individually as disfluencies and collectively as the phenomenon of repair.

Restarts

Disfluencies include the use of the words *uh* and *um*, word repetitions, **restarts**, and word fragments. The ATIS sentence in Fig. 12.19 shows examples of a restart and the use of *uh*. The restart here occurs when the speaker starts by asking for *one-way flights*. and then stops and corrects herself, restarting and asking about *one-way fares*.

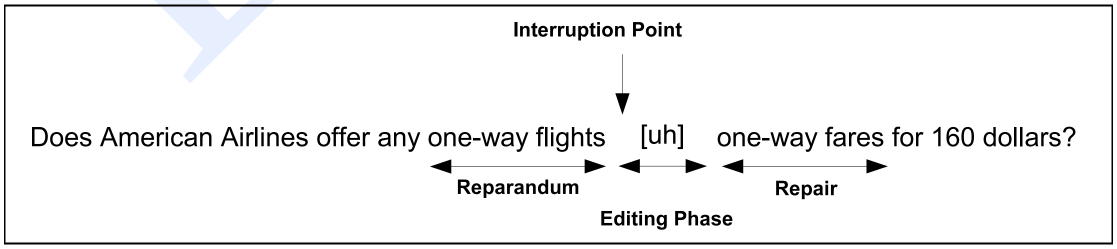

Interruption Point
Does American Airlines offer any one-way flights
[uh]
one-way fares for 160 dollars?
Reparandum
Editing Phase
Repair

Figure 12.19 An example of a disfluency (after Shriberg (1994)); terminology is from Levelt (1983)).

Reparandum
Repair
Interruption point
Edit terms

The segment *one-way flights* is referred to as the **reparandum**, and the replacing sequence *one-way fares* is referred to as the **repair**. The **interruption point**, where the speaker breaks off the original word sequence, here occurs right after the word *flights*. In the editing phase we see what are often called **edit terms**, such as *you know*, *I mean*, *uh*, and *um*.

Filled pauses

The words *uh* and *um* (sometimes called **filled pauses** or **fillers**) are generally treated like regular words in speech recognition lexicons and grammars.

Fragments

Incomplete words like *wha-* and *betwee-* in Fig. 12.18 are known as **fragments**. Fragments are extremely problematic for speech recognition systems, since they are often incorrectly attached to previous or following words, resulting in word missegmentation.

Disfluencies are very common. One count in the Switchboard Treebank corpus found that 37% of the sentences with more than two words were disfluent in some way. Indeed, the word *uh* is one of the most frequent words in Switchboard.

For applications like speech understanding, where our goal is to build a meaning for the input sentence, it may be useful to detect these restarts in order to edit out what the speaker probably considered the "corrected" words. For example in the sentence above, if we could detect that there was a restart, we could just delete the reparandum, and parse the remaining parts of the sentence:

> Does American airlines offer any $\underline{\text{one-way flights uh}}$ one-way fares for 160 dollars?

How do disfluencies interact with the constituent structure of the sentence? Hindle (1983) showed that the repair often has the same structure as the constituent just before the interruption point. Thus in the example above, the repair is an NP, as is the reparandum. This means that if it is possible to automatically find the interruption point, it is also often possible to automatically detect the boundaries of the reparandum.

There are other interactions between disfluencies and syntactic structure. For example when there is a disfluency immediately after a subject NP, the repair always repeats the subject but not the preceding discourse marker. If the repair happens after an auxiliary or main verb, the verb and subject are (almost) always recycled together (Fox and Jasperson, 1995).

12.8.2 Treebanks for Spoken Language

Treebanks for spoken corpora like Switchboard use an augmented notation to deal with spoken language phenomena like disfluencies. Fig. 12.20 shows the parse tree for Switchboard sentence (12.17). This sentence shows how the Treebank marks disfluencies; square brackets are used to separate out the entire repair area, including the reparandum, editing phase, and the repair. The plus symbol marks the end of the reparandum.

(12.17) But I don't have [any, + {F uh, } any] real idea

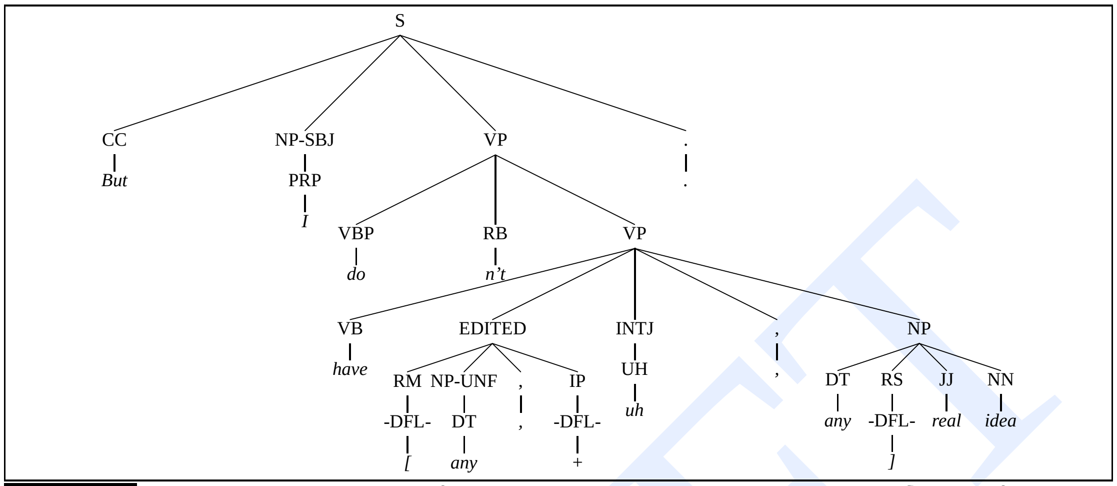


Figure 12.20 Penn Treebank III parse tree for a Switchboard sentence, showing how the disfluency information is represented in the parse tree. Note the .EDITED node, with the .RM and .RS nodes marking the beginning and end of the repair portion, and the use of the filled pause *uh*.

12.9 Grammars and Human Processing

Do people use context-free grammars in their mental processing of language? It has proved very difficult to find clear-cut evidence that they do. For example, some early experiments asked subjects to judge which words in a sentence were more closely connected (Levelt, 1970), finding that their intuitive groupings corresponded to syntactic constituents. Other experimenters examined the role of constituents in auditory comprehension by having subjects listen to sentences while also listening to short "clicks" at different times. Fodor and Bever (1965) found that subjects often mis-heard the clicks as if they occurred at constituent boundaries. They argued that the constituent was thus a "perceptual unit" which resisted interruption. Unfortunately there were severe methodological problems with the click paradigm (see e.g., Clark and Clark (1977) for a discussion).

A broader problem with all these early studies is that they do not control for the fact that constituents are often semantic units as well as syntactic units. Thus, as will be discussed further in Ch. 18, *a single odd block* is a constituent (an *NP*) but also a semantic unit (an object of type BLOCK which has certain properties). Thus experiments which show that people notice the boundaries of constituents could simply be measuring a semantic rather than a syntactic fact.

Thus it is necessary to find evidence for a constituent which is *not* a semantic unit. Furthermore, since there are many non-constituent-based theories of grammar based on lexical dependencies, it is important to find evidence that cannot be interpreted as a *lexical* fact; that is, evidence for constituency that is not based on particular words.

One series of experiments arguing for constituency has come from Kathryn Bock and her colleagues. Bock and Loebell (1990), for example, avoided all these earlier pitfalls by studying whether a subject who uses a particular syntactic constituent (e.g., a verb-phrase of a particular type, like *V NP PP*), is more likely to use the constituent in following sentences. In other words, they asked whether use of a constituent *primes* its use in subsequent sentences. As we saw in previous chapters, priming is a common way to test for the existence of a mental structure. Bock and Loebell relied on the English **ditransitive alternation**. A ditransitive verb is one like *give* which can take two arguments:

(12.18) The wealthy widow gave [$_{NP}$ the church] [$_{NP}$ her Mercedes].

The verb *give* allows another possible subcategorization frame, called a **prepositional dative** in which the indirect object is expressed as a prepositional phrase:

(12.19) The wealthy widow gave [$_{NP}$ her Mercedes] [$_{PP}$ to the church].

Alternations

As we discussed on page 406, many verbs other than *give* have such **alternations** (*send*, *sell*, etc.; see Levin (1993) for a summary of many different alternation patterns). Bock and Loebell relied on these alternations by giving subjects a picture, and asking them to describe it in one sentence. The picture was designed to elicit verbs like *give* or *sell* by showing an event such as a boy handing an apple to a teacher. Since these verbs alternate, subjects might, for example, say *The boy gave the apple to the teacher* or *The boy gave the teacher an apple*.

Before describing the picture, subjects were asked to read an unrelated “priming” sentence out loud; the priming sentences either had *V NP NP* or *V NP PP* structure. Crucially, while these priming sentences had the same *constituent structure* as the dative alternation sentences, they did not have the same *semantics*. For example, the priming sentences might be prepositional *locatives*, rather than *datives*:

(12.20) IBM moved [$_{NP}$ a bigger computer] [$_{PP}$ to the Sears store].

Bock and Loebell found that subjects who had just read a *V NP PP* sentence were more likely to use a *V NP PP* structure in describing the picture. This suggested that the use of a particular constituent *primed* the later use of that constituent, and hence that the constituent must be mentally represented in order to prime and be primed.

In more recent work, Bock and her colleagues have continued to find evidence for this kind of constituency structure.

12.10 Summary

This chapter has introduced a number of fundamental concepts in syntax via the **context-free grammar**.

- In many languages, groups of consecutive words act as a group or a **constituent**, which can be modeled by **context-free grammars** (also known as **phrase-structure grammars**).
- A context-free grammar consists of a set of **rules** or **productions**, expressed over a set of **non-terminal** symbols and a set of **terminal** symbols. Formally, a particular **context-free language** is the set of strings which can be **derived** from a particular **context-free grammar**.

- A **generative grammar** is a traditional name in linguistics for a formal language which is used to model the grammar of a natural language.
- There are many sentence-level grammatical constructions in English; **declarative**, **imperative**, **yes-no-question**, and **wh-question** are four very common types, which can be modeled with context-free rules.
- An English **noun phrase** can have **determiners**, **numbers**, **quantifiers**, and **adjective phrases** preceding the **head noun**, which can be followed by a number of **postmodifiers**; **gerundive** VPs, **infinitives** VPs, and **past participial** VPs are common possibilities.
- **Subjects** in English **agree** with the main verb in person and number.
- Verbs can be **subcategorized** by the types of **complements** they expect. Simple subcategories are **transitive** and **intransitive**; most grammars include many more categories than these.
- The correlate of **sentences** in spoken language are generally called **utterances**. Utterances may be **disfluent**, containing **filled pauses** like *um* and *uh*, **restarts**, and **repairs**.
- **Treebanks** of parsed sentences exist for many genres of English and for many languages. Treebanks can be searched using tree-search tools.
- Any context-free grammar can be converted to **Chomsky normal form**, in which the right-hand-side of each rule has either two non-terminals or a single terminal.
- Context-free grammars are more powerful than finite-state automata, but it is nonetheless possible to **approximate** a context-free grammar with a FSA.
- There is some evidence that constituency plays a role in the human processing of language.

Bibliographical and Historical Notes

“den sprachlichen Ausdruck für die willkürliche Gliederung einer Gesammtvorstellung in ihre in logische Beziehung zueinander gesetzten Bestandteile”
“the linguistic expression for the arbitrary division of a total idea into its constituent parts placed in logical relations to one another”

Wundt’s (1900:240) definition of the sentence; the origin of the idea of phrasal constituency, cited in Percival (1976).

According to Percival (1976), the idea of breaking up a sentence into a hierarchy of constituents appeared in the *Völkerpsychologie* of the groundbreaking psychologist Wilhelm Wundt (Wundt, 1900). Wundt’s idea of constituency was taken up into linguistics by Leonard Bloomfield in his early book *An Introduction to the Study of Language* (Bloomfield, 1914). By the time of his later book *Language* (Bloomfield, 1933), what was then called “immediate-constituent analysis” was a well-established method of syntactic study in the United States. By contrast, traditional European grammar, dating from the Classical period, defined relations between *words* rather than constituents, and European syntacticians retained this emphasis on such **dependency** grammars.

American Structuralism saw a number of specific definitions of the immediate constituent, couched in terms of their search for a “discovery procedure”; a methodological algorithm for describing the syntax of a language. In general, these attempt to capture the intuition that “The primary criterion of the immediate constituent is the degree in which combinations behave as simple units” (Bazell, 1966, p. 284). The most well-known of the specific definitions is Harris’ idea of distributional similarity to individual units, with the *substitutability* test. Essentially, the method proceeded by breaking up a construction into constituents by attempting to substitute simple structures for possible constituents—if a substitution of a simple form, say *man*, was substitutable in a construction for a more complex set (like *intense young man*), then the form *intense young man* was probably a constituent. Harris’s test was the beginning of the intuition that a constituent is a kind of equivalence class.

The first formalization of this idea of hierarchical constituency was the **phrase-structure grammar** defined in Chomsky (1956), and further expanded upon (and argued against) in Chomsky (1957) and Chomsky (1975). From this time on, most generative linguistic theories were based at least in part on context-free grammars or generalizations of them (such as Head-Driven Phrase Structure Grammar (Pollard and Sag, 1994), Lexical-Functional Grammar (Bresnan, 1982), Government and Binding (Chomsky, 1981), and Construction Grammar (Kay and Fillmore, 1999), inter alia); many of these theories used schematic context-free templates known as **X-bar schemata** which also relied on the notion of syntactic head.

X-bar schemata

Shortly after Chomsky’s initial work, the context-free grammar was rediscovered by Backus (1959) and independently by Naur et al. (1960) in their descriptions of the ALGOL programming language; Backus (1996) noted that he was influenced by the productions of Emil Post and that Naur’s work was independent of his (Backus’) own. (Recall the discussion on page 13 of multiple invention in science.) After this early work, a great number of computational models of natural language processing were based on context-free grammars because of the early development of efficient algorithms to parse these grammars (see Ch. 13).

As we have already noted, grammars based on context-free rules are not ubiquitous. Various classes of extensions to CFGs are designed specifically to handle long-distance dependencies. We noted earlier that some grammars treat long-distance-dependent items as being related semantically but not syntactically; the surface syntax does not represent the long-distance link (Kay and Fillmore, 1999; Culicover and Jackendoff, 2005). But there are alternatives. One extended formalism is **Tree Adjoining Grammar** (TAG) (Joshi, 1985). The primary data structure in Tree Adjoining Grammar is the tree, rather than the rule. Trees come in two kinds; **initial trees** and **auxiliary trees**. Initial trees might, for example, represent simple sentential structures, while auxiliary trees are used to add recursion into a tree. Trees are combined by two operations called **substitution** and **adjunction**. The adjunction operation is used to handle long-distance dependencies. See Joshi (1985) for more details. An extension of Tree Adjoining Grammar called Lexicalized Tree Adjoining Grammars will be discussed in Ch. 14. Tree Adjoining Grammar is a member of the family of **mildly context-sensitive languages** to be introduced in Ch. 15.

We mentioned on page 408 another way of handling long-distance dependencies, based on the use of empty categories and co-indexing. The Penn Treebank uses this model, which draws (in various Treebank corpora) from the Extended Standard Theory and Minimalism (Radford, 1997).

Representative examples of grammars that are based on word relations rather than constituency include the dependency grammar of Mel'čuk (1979), the Word Grammar of Hudson (1984), and the Constraint Grammar of Karlsson et al. (1995b).

There are a variety of algorithms for building a regular grammar which approximates a CFG (Pereira and Wright, 1997; Johnson, 1998a; Langendoen and Langsam, 1987; Nederhof, 2000; Mohri and Nederhof, 2001).

Readers interested in the grammar of English should get one of the three large reference grammars of English: Huddleston and Pullum (2002), Biber et al. (1999), and Quirk et al. (1985), Another useful reference is McCawley (1998).

generative

functional

There are many good introductory textbooks on syntax from different perspectives. Sag et al. (2003) is an introduction to syntax from a **generative** perspective, focusing on the use of phrase-structure, unification, and the type-hierarchy in Head-Driven Phrase Structure Grammar. Van Valin and La Polla (1997) is an introduction from a **functional** perspective, focusing on cross-linguistic data and on the functional motivation for syntactic structures.

See Bach (1988) for an introduction to basic categorial grammar. Various extensions to categorial grammars are presented in Lambek (1958), Dowty (1979), and Ades and Steedman (1982) inter alia; the other papers in Oehrle et al. (1988) give a survey of extensions. Combinatory categorial grammar is presented in Steedman (1989, 2000); see Steedman and Baldridge (2007) for a tutorial introduction. See Ch. 18 for a discussion of semantic composition.

Exercises

12.1 Draw tree structures for the following ATIS phrases:

a. Dallas
b. from Denver
c. after five p.m.
d. arriving in Washington
e. early flights
f. all redeye flights
g. on Thursday
h. a one-way fare
i. any delays in Denver

12.2 Draw tree structures for the following ATIS sentences:

a. Does American airlines have a flight between five a.m. and six a.m.

b. I would like to fly on American airlines.
c. Please repeat that.
d. Does American 487 have a first class section?
e. I need to fly between Philadelphia and Atlanta.
f. What is the fare from Atlanta to Denver?
g. Is there an American airlines flight from Philadelphia to Dallas?

12.3 Augment the grammar rules on page 403 to handle pronouns. Deal properly with person and case.

12.4 Modify the noun phrase grammar of Sections 12.3.3–12.3.4 to correctly model mass nouns and their agreement properties

12.5 How many types of *NP*s would the rule on page 400 expand to if we didn't allow parentheses in our grammar formalism?

12.6 Assume a grammar that has many *VP* rules for different subcategorizations, as expressed in Sec. 12.3.5, and differently subcategorized verb rules like *Verb-with-NP-complement*. How would the rule for post-nominal relative clauses (12.7) need to be modified if we wanted to deal properly with examples like *the earliest flight that you have*? Recall that in such examples the pronoun *that* is the object of the verb *get*. Your rules should allow this noun phrase but should correctly rule out the ungrammatical S **I get*.

12.7 Does your solution to the previous problem correctly model the NP *the earliest flight that I can get*? How about *the earliest flight that I think my mother wants me to book for her*? Hint: this phenomenon is called **long-distance dependency**.

12.8 Write rules expressing the verbal subcategory of English auxiliaries; for example you might have a rule *verb-with-bare-stem-VP-complement* → *can*.

possessive
genitive

12.9 *NP*s like *Fortune's office* or *my uncle's marks* are called **possessive** or **genitive** noun phrases. A possessive noun phrase can be modeled by treating the sub-NP like *Fortune's* or *my uncle's* as a determiner of the following head noun. Write grammar rules for English possessives. You may treat *'s* as if it were a separate word (i.e., as if there were always a space before *'s*).

12.10 Page 397 discussed the need for a *Wh-NP* constituent. The simplest *Wh-NP* is one of the *Wh-pronouns* (*who, whom, whose, which*). The Wh-words *what* and *which* can be determiners: *which four will you have?*, *what credit do you have with the Duke?* Write rules for the different types of *Wh-NP*s.

12.11 Write an algorithm for converting an arbitrary context-free grammar into Chomsky normal form.