CHAPTER

12 Constituency Grammars

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)

syntax

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. And our 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: ordering of sequences of words (Chapter 2), probabilities for these word sequences (Chapter 3), and the use of part-of-speech categories as a grammatical equivalence class for words (Chapter 8). In this chapter and the next three we introduce a variety of syntactic phenomena that go well beyond these simpler approaches, together with formal models for capturing them in a computationally useful manner.

The bulk of this chapter is devoted to context-free grammars. Context-free grammars are the backbone of many formal models of the syntax of natural language (and, for that matter, of computer languages). As such, they play a role in 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 show in Chapter 13). And in Chapter 16 we show how they provide a systematic framework for semantic interpretation. Here we also introduce the concept of lexicalized grammars, focusing on one example, **combinatory categorial grammar**, or **CCG**.

In Chapter 14 we introduce a formal model of grammar called **syntactic dependencies** that is an alternative to these constituency grammars, and we'll give algorithms for **dependency parsing**. Both constituency and dependency formalisms are important for language processing.

Finally, we provide a brief overview of the grammar of English, illustrated from a domain with relatively simple sentences called ATIS (Air Traffic Information System) (Hemphill et al., 1990). ATIS systems were an early spoken language system for users to book flights, by expressing sentences like *I'd like to fly to Atlanta*.

12.1 Constituency

Syntactic constituency is the idea that groups of words can behave as single units, or constituents. Part of developing a grammar involves building an inventory of the constituents in the language. 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):

noun phrase

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

What evidence do we have 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 *sat...*

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 at the beginning (preposed) or at the end (postposed):

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

12.2 Context-Free Grammars

CFG

The most widely used formal 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 **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

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

lexicon

NP

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

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

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

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 abstractions over these terminals are called **non-terminals**. In each context-free rule, the item to the right of the arrow ($\rightarrow$) is an ordered list of one or more terminals and non-terminals; to the left of the arrow is a single non-terminal symbol expressing some cluster or generalization. The non-terminal associated with each word in the lexicon is its lexical category, or part of speech.

terminal

non-terminal

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. Viewing a CFG as a generator, we can read the $\rightarrow$ 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 our first rule to rewrite *NP* as:	*Det Nominal*
and then rewrite *Nominal* as:	*Det Noun*
and finally rewrite these parts-of-speech as:	*a flight*

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). Figure 12.1 shows the tree representation of this derivation.

derivation

parse tree

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.

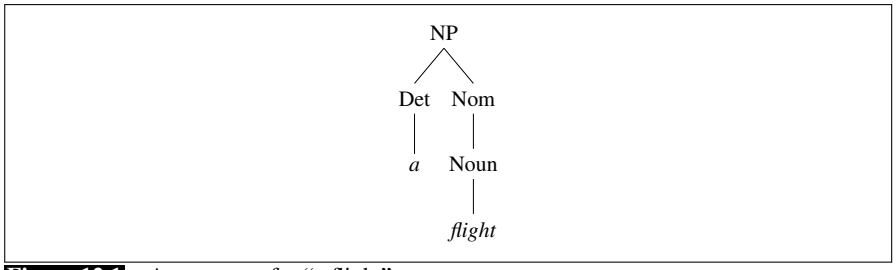


Figure 12.1 A parse tree for "a flight".

verb phrase

Let's add a few additional rules to our inventory. The following rule expresses 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 may be followed by a noun phrase and a prepositional phrase:

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

Or the verb phrase may have a verb 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 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

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

bracketed notation

We can also represent a parse tree in a more compact format called **bracketed notation**; here is the bracketed representation of the parse tree of Fig. 12.4:

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.

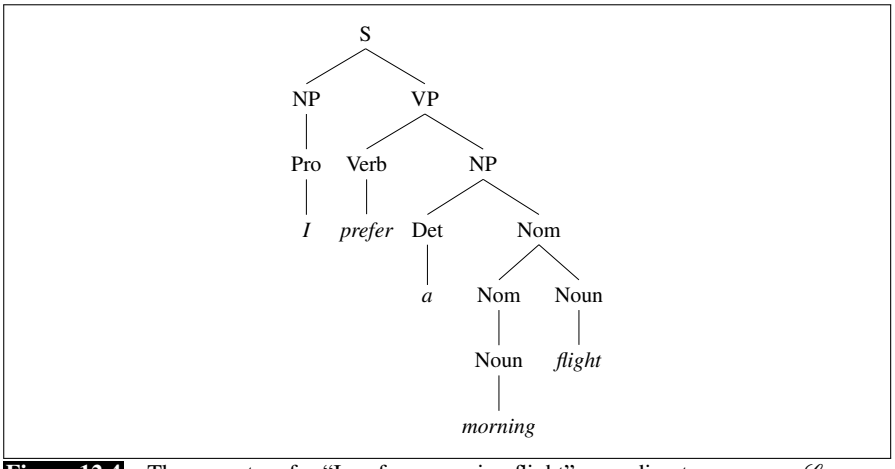


Figure 12.4 The parse tree for "I prefer a morning flight" according to grammar $\mathscr{L}_0$.

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

A CFG like that of $\mathscr{L}_0$ defines a formal language. We saw in Chapter 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**.

grammatical
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.

generative grammar

12.2.1 Formal Definition of Context-Free Grammar

We conclude this section 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 this 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** and a member of N

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

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

directly derives

> if $A \rightarrow \beta$ is a production of R 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
>
> $$\alpha_1 \Rightarrow \alpha_2, \alpha_2 \Rightarrow \alpha_3, \ldots, \alpha_{m-1} \Rightarrow \alpha_m$$

derives

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

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

$$\mathscr{L}_G = \{w | w \text{ is in } \Sigma* \text{ and } S \stackrel{*}{\Rightarrow} w\}$$

syntactic parsing

The problem of mapping from a string of words to its parse tree is called **syntactic parsing**; we define algorithms for constituency parsing in Chapter 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 is necessarily 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*. Among the large number of constructions for English sentences, four are particularly common and important: declaratives, imperatives, yes-no questions, and wh-questions.

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 follow up on in Chapter 24. Here are a number of examples from the ATIS domain:

I want a flight from Ontario to Chicago
The flight should be eleven a.m. tomorrow
The return flight should leave at around seven p.m.

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
Give me Sunday's flights arriving in Las Vegas from New York City
List all flights between five and seven p.m.

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; they 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 a question at all but a request; Chapter 24 discusses 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 sentence-level structures we examine here 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?

Here is a rule. Exercise 12.7 discusses rules for the constituents that make up the *Wh-NP*.

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

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?

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

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. Other models of parsing represent the relationship between *flights* and *have* as a syntactic relation, and the grammar is modified to insert a small marker called a **trace** or **empty category** after the verb. We discuss empty-category models when we introduce the Penn Treebank on page 15.

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, *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* than just standing alone as a unit of discourse.

clause

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**, which traditional grammars often describe 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 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 on page 5. 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.

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. The central focus of this section is on the last type since that is where the bulk of the syntactic complexity resides. These noun phrases consist 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:

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

The role of the determiner can also be filled by more complex expressions:

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*) helps us model the last two examples above, in which a sequence of possessive expressions serves as a determiner.

Under some circumstances determiners are optional in English. For example, determiners may be omitted if the noun they modify is plural:

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

As we saw in Chapter 8, **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*:

(12.3) Does this flight serve dinner?

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 but after the determiner (the "postdeterminers") in a nominal. These include **cardinal numbers**, **ordinal numbers**, **quantifiers**, and adjectives. 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

Adjectives occur after quantifiers but before nouns.

a *first-class* fare	a *non-stop* 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 Chapter 8 for definitions of adjectives and adverbs):

the *least expensive* fare

After the Head Noun

A head noun can be followed by **postmodifiers**. Three kinds of nominal postmodifiers are 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*

They are especially common in the ATIS corpus since they are used to mark the origin and destination of flights.

Here are some examples of prepositional phrase postmodifiers, with brackets inserted to show the boundaries of each PP; note that two or more PPs can be strung together within a single NP:

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. Here are some examples:

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*?

relative pronoun

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 in the following examples:

a flight *that serves breakfast*
flights *that leave in the morning*
the one *that leaves at ten thirty five*

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

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

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

the earliest American Airlines flight that I can get

Various postnominal modifiers can be combined:

a flight *[from Phoenix to Detroit] [leaving Monday evening]*
evening flights *[from Nashville to Houston] [that serve dinner]*
a friend *[living in Denver] [that would like to visit me in 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 The Verb Phrase

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:

$$\begin{aligned} \textit{VP} &\rightarrow \textit{Verb} && \text{disappear} \\ \textit{VP} &\rightarrow \textit{Verb NP} && \text{prefer a morning flight} \\ \textit{VP} &\rightarrow \textit{Verb NP PP} && \text{leave Boston in the morning} \\ \textit{VP} &\rightarrow \textit{Verb PP} && \text{leaving on Thursday} \end{aligned}$$

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

sentential complements

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 to downtown]]
I $[_{VP}$ $[_{V}$ think $[_{S}$ I would like to take the nine thirty flight]]

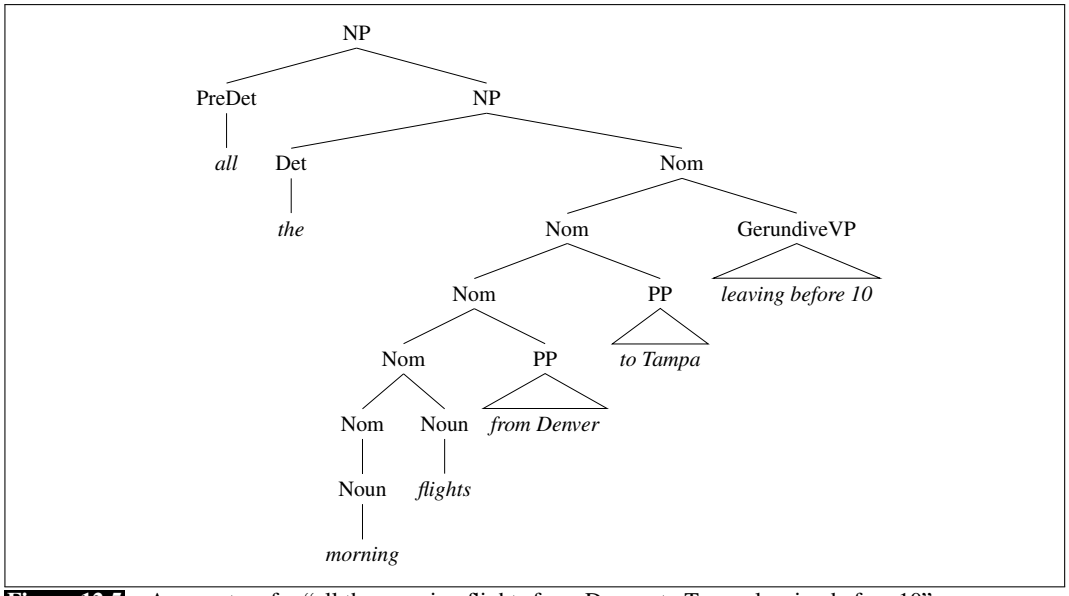


Figure 12.5 A parse tree for "all the morning flights from Denver to Tampa leaving before 10".

Here's a rule for these:

$$VP \rightarrow Verb\ S$$

Similarly, 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]

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 be used either 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*).

transitive
intransitive

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*).

subcategorize
subcategorizes for
complements
subcategorization frame

Where traditional grammars **subcategorize** verbs into these two categories (transitive and intransitive), modern grammars distinguish as many as 100 subcategories. We say that a verb like *find* **subcategorizes for** an *NP*, and 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 talk more about this view of verbs and arguments in Chapter 15 when we talk about predicate calculus representations of verb semantics. Subcategorization frames for a set of example verbs are given in Fig. 12.6.

Frame	Verb	Example
$\emptyset$	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]
S	mean	Does this mean [$_{S}$ AA has a hub in Boston]

Figure 12.6 Subcategorization frames for a set of example verbs.

We can capture the association between verbs and their complements by making separate subtypes of the class Verb (e.g., *Verb-with-NP-complement*, *Verb-with-Inf-VP-complement*, *Verb-with-S-complement*, and so on):

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

Each *VP* rule could then be modified to require the appropriate verb subtype:

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

A problem with this approach is the significant increase in the number of rules and the associated loss of generality.

12.3.5 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:

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

Note that the ability to form coordinate phrases through 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 rule for this:

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

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.

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

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 GPSG (Gazdar et al., 1985) do this using **metarules** like:

metarules

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

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

12.4 Treebanks

Sufficiently robust grammars consisting of context-free grammar rules can be used to assign a parse tree to any sentence. This means that it is possible to build a corpus where every sentence in the collection is paired with a corresponding parse tree. Such a syntactically annotated corpus is called a **treebank**. Treebanks play an important role in parsing, as we discuss in Chapter 13, as well as in linguistic investigations of syntactic phenomena.

treebank

A wide variety of treebanks have been created, generally through the use of parsers (of the sort described in the next few chapters) to automatically parse each sentence, followed by the use of humans (linguists) to hand-correct the parses. The **Penn Treebank** project (whose POS tagset we introduced in Chapter 8) has produced treebanks from the Brown, Switchboard, ATIS, and *Wall Street Journal* corpora of English, as well as treebanks in Arabic and Chinese. A number of treebanks use the dependency representation we will introduce in Chapter 14, including many that are part of the **Universal Dependencies** project (Nivre et al., 2016).

Penn Treebank

12.4.1 Example: The Penn Treebank Project

Figure 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 Chapter 8. The use of LISP-style parenthesized notation for trees is extremely common and resembles the bracketed notation we saw earlier in (12.1). For those who are not familiar with it we show a standard node-and-line tree representation in Fig. 12.8.

Figure 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

traces

[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 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 (a) Brown and (b) ATIS corpora.

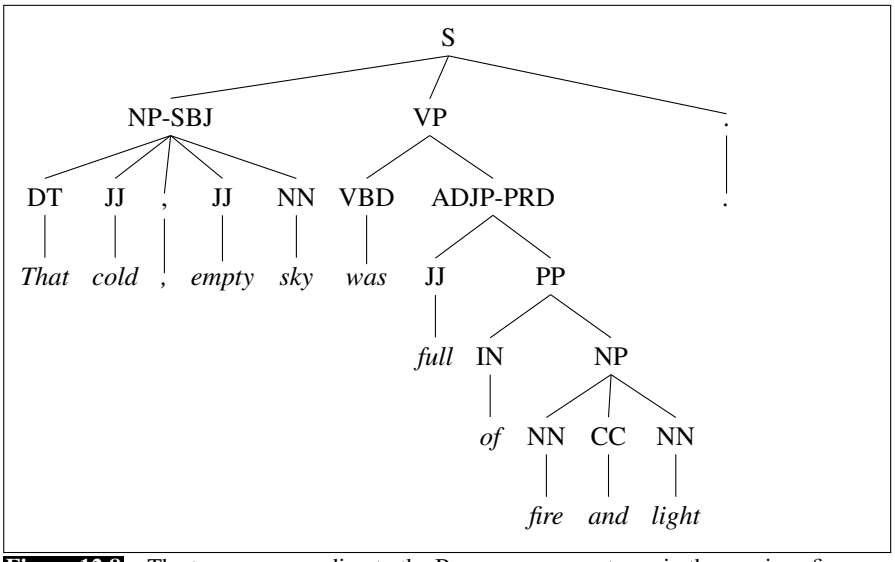


Figure 12.8 The tree corresponding to the Brown corpus sentence in the previous figure.

syntactic movement

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 the node `-NONE-` marks 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 co-indexing 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 marks 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 co-indexed 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) its presence in particular text categories (headlines, titles), and its semantic function (temporal phrases, lo-

```
( (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.

cations) (Marcus et al. 1994, Bies et al. 1995). Figure 12.9 shows examples of the -SBJ (surface subject) and -TMP (temporal phrase) tags. Figure 12.8 shows in addition the -PRD tag, which is used for predicates that are not VPs (the one in Fig. 12.8 is an ADJP). We'll return to the topic of grammatical function when we consider dependency grammars and parsing in Chapter 14.

12.4.2 Treebanks as Grammars

The sentences in a treebank implicitly constitute a grammar of the language represented by the corpus being annotated. For example, from the three parsed sentences in Fig. 12.7 and Fig. 12.9, we can 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 VPs 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
VP → ADVP VB PP
```

as well as even longer rules, such as

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

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

which comes from the *VP* marked in italics:

> 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 noun phrases:

[DT The] [JJ state-owned] [JJ industrial] [VBG holding] [NN company] [NNP Instituto] [NNP Nacional] [FW de] [NNP Industria]

[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.

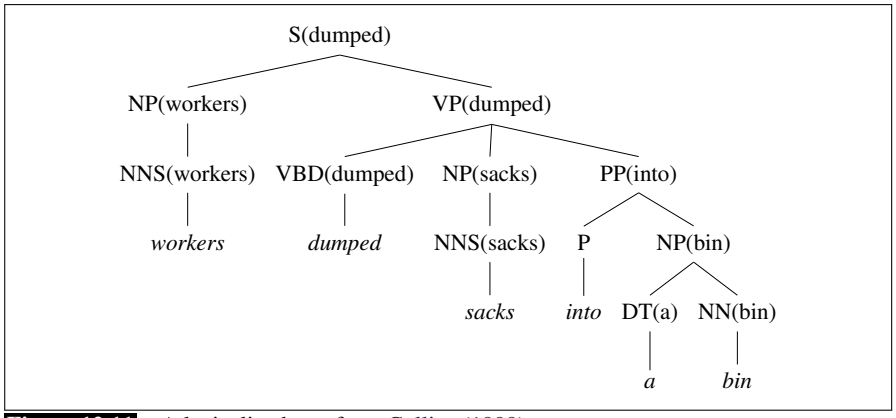


Figure 12.11 A lexicalized tree from Collins (1999).

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 discuss these further in Appendix C.

12.4.3 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), and is central to the dependency grammars and dependency parsing we'll introduce in Chapter 14. Heads are also important in probabilistic parsing (Appendix C) and in constituent-based grammar formalisms like Head-Driven Phrase Structure Grammar (Pollard and Sag, 1994)..

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 that 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. Figure 12.11 shows an example of such a tree from Collins (1999), in which each non-terminal is annotated with its head.

For the generation of such a tree, each CFG rule must be augmented to identify one right-side constituent to be the head child. The headword for a node is then set to the headword of its head child. Choosing these head children 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 finding a head 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 handwritten rules, such as a practical one for Penn Treebank grammars given in Collins (1999) but developed originally by Magerman (1995). For example, the rule for finding the head of an *NP* is as follows (Collins, 1999, p. 238):

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

- 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 this set are shown in Fig. 12.12. 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 VBN, 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 12.12 Some head rules from Collins (1999). The head rules are also 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**, that is they have binary trees (down to the prelexical nodes). We make use of this binary branching property in the CKY parsing algorithm in Chapter 13.

Chomsky normal form

binary branching

Any context-free 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 12.8 asks the reader to formulate the complete algorithm):

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

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

```
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

The generation of a symbol A with a potentially infinite sequence of symbols B with a rule of the form `A → A B` is known as **Chomsky-adjunction**.

12.6 Lexicalized Grammars

The approach to grammar presented thus far emphasizes phrase-structure rules while minimizing the role of the lexicon. However, as we saw in the discussions of agreement, subcategorization, and long-distance dependencies, this approach leads to solutions that are cumbersome at best, yielding grammars that are redundant, hard to manage, and brittle. To overcome these issues, numerous alternative approaches have been developed that all share the common theme of making better use of the lexicon. Among the more computationally relevant approaches are Lexical-Functional Grammar (LFG) (Bresnan, 1982), Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994), Tree-Adjoining Grammar (TAG) (Joshi, 1985), and Combinatory Categorial Grammar (CCG). These approaches differ with respect to how *lexicalized* they are — the degree to which they rely on the lexicon as opposed to phrase structure rules to capture facts about the language.

The following section provides an introduction to CCG, a heavily lexicalized approach motivated by both syntactic and semantic considerations, which we will return to in Chapter 15. Chapter 14 discusses dependency grammars, an approach that eliminates phrase-structure rules entirely.

12.6.1 Combinatory Categorial Grammar

categorial grammar

combinatory categorial grammar

In this section, we provide an overview of **categorial grammar** (Ajdukiewicz 1935, Bar-Hillel 1953), an early lexicalized grammar model, as well as an important modern extension, **combinatory categorial grammar**, or CCG (Steedman 1996, Steedman 1989, Steedman 2000).

The categorial approach consists of three major elements: a set of categories, a lexicon that associates words with categories, and a set of rules that govern how categories combine in context.

Categories

Categories are either atomic elements or single-argument functions that return a category as a value when provided with a desired category as argument. More formally, we can define $\mathscr{C}$, a set of categories for a grammar as follows:

- $\mathscr{A} \subseteq \mathscr{C}$, where $\mathscr{A}$ is a given set of atomic elements
- (X/Y), $(X\backslash Y) \in \mathscr{C}$, if X, $Y \in \mathscr{C}$

The slash notation shown here is used to define the functions in the grammar. It specifies the type of the expected argument, the direction it is expected be found, and the type of the result. Thus, (X/Y) is a function that seeks a constituent of type Y to its right and returns a value of X; $(X\backslash Y)$ is the same except it seeks its argument to the left.

The set of atomic categories is typically very small and includes familiar elements such as sentences and noun phrases. Functional categories include verb phrases and complex noun phrases among others.

The Lexicon

The lexicon in a categorial approach consists of assignments of categories to words. These assignments can either be to atomic or functional categories, and due to lexical ambiguity words can be assigned to multiple categories. Consider the following sample lexical entries.

$$\begin{aligned} \textit{flight}: &\quad N \\ \textit{Miami}: &\quad NP \\ \textit{cancel}: &\quad (S\backslash NP)/NP \end{aligned}$$

Nouns and proper nouns like *flight* and *Miami* are assigned to atomic categories, reflecting their typical role as arguments to functions. On the other hand, a transitive verb like *cancel* is assigned the category $(S\backslash NP)/NP$: a function that seeks an NP on its right and returns as its value a function with the type $(S\backslash NP)$. This function can, in turn, combine with an NP on the left, yielding an S as the result. This captures the kind of subcategorization information discussed in Section 12.3.4, however here the information has a rich, computationally useful, internal structure.

Ditransitive verbs like *give*, which expect two arguments after the verb, would have the category $((S\backslash NP)/NP)/NP$: a function that combines with an NP on its right to yield yet another function corresponding to the transitive verb $(S\backslash NP)/NP$ category such as the one given above for *cancel*.

Rules

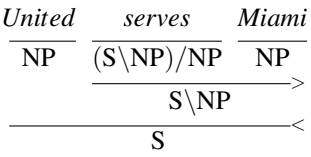
The rules of a categorial grammar specify how functions and their arguments combine. The following two rule templates constitute the basis for all categorial grammars.

$$X/Y \;\; Y \;\Rightarrow\; X \tag{12.4}$$

$$Y \;\; X\backslash Y \;\Rightarrow\; X \tag{12.5}$$

The first rule applies a function to its argument on the right, while the second looks to the left for its argument. We'll refer to the first as **forward function application**, and the second as **backward function application**. The result of applying either of these rules is the category specified as the value of the function being applied.

Given these rules and a simple lexicon, let's consider an analysis of the sentence *United serves Miami*. Assume that *serves* is a transitive verb with the category $(S\backslash NP)/NP$ and that *United* and *Miami* are both simple *NP*s. Using both forward and backward function application, the derivation would proceed as follows:

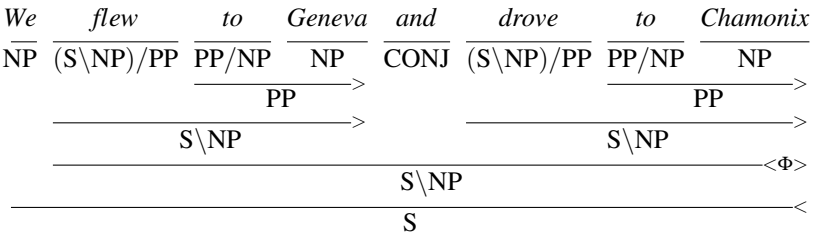


Categorial grammar derivations are illustrated growing down from the words, rule applications are illustrated with a horizontal line that spans the elements involved, with the type of the operation indicated at the right end of the line. In this example, there are two function applications: one forward function application indicated by the $>$ that applies the verb *serves* to the *NP* on its right, and one backward function application indicated by the $<$ that applies the result of the first to the *NP* *United* on its left.

With the addition of another rule, the categorial approach provides a straightforward way to implement the coordination metarule described earlier on page 14. Recall that English permits the coordination of two constituents of the same type, resulting in a new constituent of the same type. The following rule provides the mechanism to handle such examples.

$$X \ CONJ \ X \ \Rightarrow \ X \tag{12.6}$$

This rule states that when two constituents of the same category are separated by a constituent of type *CONJ* they can be combined into a single larger constituent of the same type. The following derivation illustrates the use of this rule.



Here the two $S\backslash NP$ constituents are combined via the conjunction operator $<\Phi>$ to form a larger constituent of the same type, which can then be combined with the subject *NP* via backward function application.

These examples illustrate the lexical nature of the categorial grammar approach. The grammatical facts about a language are largely encoded in the lexicon, while the rules of the grammar are boiled down to a set of three rules. Unfortunately, the basic categorial approach does not give us any more expressive power than we had with traditional CFG rules; it just moves information from the grammar to the lexicon. To move beyond these limitations CCG includes operations that operate over functions.

The first pair of operators permit us to **compose** adjacent functions.

$$X/Y \; Y/Z \;\Rightarrow\; X/Z \tag{12.7}$$

$$Y\backslash Z \; X\backslash Y \;\Rightarrow\; X\backslash Z \tag{12.8}$$

forward composition

The first rule, called **forward composition**, can be applied to adjacent constituents where the first is a function seeking an argument of type Y to its right, and the second is a function that provides Y as a result. This rule allows us to compose these two functions into a single one with the type of the first constituent and the argument of the second. Although the notation is a little awkward, the second rule, **backward composition** is the same, except that we're looking to the left instead of to the right for the relevant arguments. Both kinds of composition are signalled by a **B** in CCG diagrams, accompanied by a $<$ or $>$ to indicate the direction.

backward composition

type raising

The next operator is **type raising**. Type raising elevates simple categories to the status of functions. More specifically, type raising takes a category and converts it to function that seeks as an argument a function that takes the original category as its argument. The following schema show two versions of type raising: one for arguments to the right, and one for the left.

$$X \;\Rightarrow\; T/(T\backslash X) \tag{12.9}$$

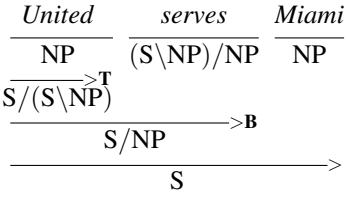
$$X \;\Rightarrow\; T\backslash(T/X) \tag{12.10}$$

The category T in these rules can correspond to any of the atomic or functional categories already present in the grammar.

A particularly useful example of type raising transforms a simple *NP* argument in subject position to a function that can compose with a following *VP*. To see how this works, let's revisit our earlier example of *United serves Miami*. Instead of classifying *United* as an *NP* which can serve as an argument to the function attached to *serve*, we can use type raising to reinvent it as a function in its own right as follows.

$$NP \;\Rightarrow\; S/(S\backslash NP)$$

Combining this type-raised constituent with the forward composition rule (12.7) permits the following alternative to our previous derivation.

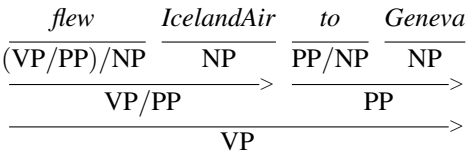


By type raising *United* to $S/(S\backslash NP)$, we can compose it with the transitive verb *serves* to yield the (S/NP) function needed to complete the derivation.

There are several interesting things to note about this derivation. First, it provides a left-to-right, word-by-word derivation that more closely mirrors the way humans process language. This makes CCG a particularly apt framework for psycholinguistic studies. Second, this derivation involves the use of an intermediate unit of analysis, *United serves*, that does not correspond to a traditional constituent in English. This ability to make use of such non-constituent elements provides CCG with the ability to handle the coordination of phrases that are not proper constituents, as in the following example.

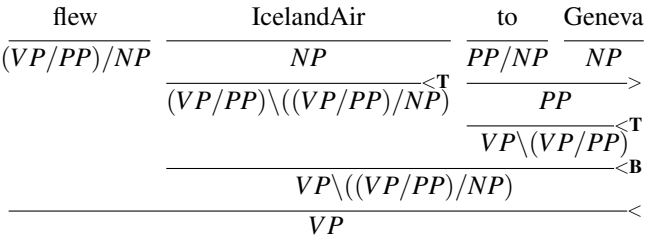
(12.11) We flew IcelandAir to Geneva and SwissAir to London.

Here, the segments that are being coordinated are *IcelandAir to Geneva* and *SwissAir to London*, phrases that would not normally be considered constituents, as can be seen in the following standard derivation for the verb phrase *flew IcelandAir to Geneva*.

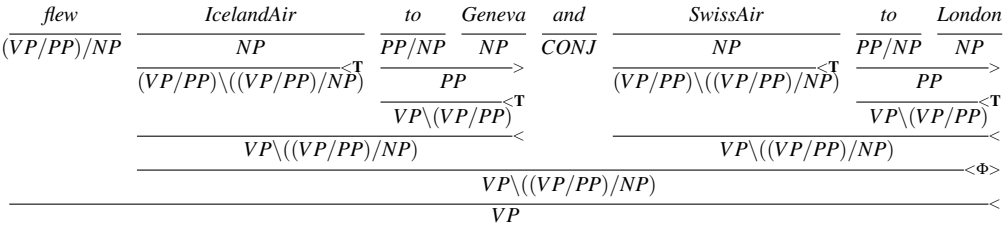


In this derivation, there is no single constituent that corresponds to *IcelandAir to Geneva*, and hence no opportunity to make use of the <Φ> operator. Note that complex CCG categories can get a little cumbersome, so we'll use *VP* as a shorthand for $(S\backslash NP)$ in this and the following derivations.

The following alternative derivation provides the required element through the use of both backward type raising (12.10) and backward function composition (12.8).



Applying the same analysis to *SwissAir to London* satisfies the requirements for the <Φ> operator, yielding the following derivation for our original example (12.11).

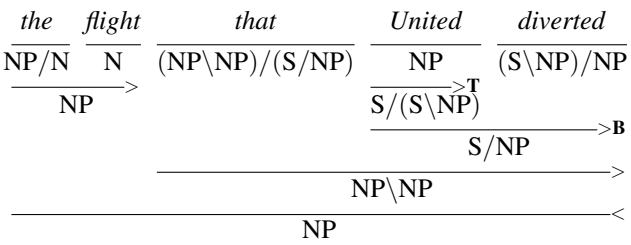


Finally, let's examine how these advanced operators can be used to handle **long-distance dependencies** (also referred to as syntactic movement or extraction). As mentioned in Section 12.3.1, long-distance dependencies arise from many English constructions including wh-questions, relative clauses, and topicalization. What these constructions have in common is a constituent that appears somewhere distant from its usual, or expected, location. Consider the following relative clause as an example.

the flight that United diverted

Here, *divert* is a transitive verb that expects two *NP* arguments, a subject *NP* to its left and a direct object *NP* to its right; its category is therefore $(S\backslash NP)/NP$. However, in this example the direct object *the flight* has been "moved" to the beginning of the clause, while the subject *United* remains in its normal position. What is needed is a way to incorporate the subject argument, while dealing with the fact that *the flight* is not in its expected location.

The following derivation accomplishes this, again through the combined use of type raising and function composition.



As we saw with our earlier examples, the first step of this derivation is type raising *United* to the category $S/(S\backslash NP)$ allowing it to combine with *diverted* via forward composition. The result of this composition is S/NP which preserves the fact that we are still looking for an *NP* to fill the missing direct object. The second critical piece is the lexical category assigned to the word *that*: $(NP\backslash NP)/(S/NP)$. This function seeks a verb phrase missing an argument to its right, and transforms it into an *NP* seeking a missing element to its left, precisely where we find *the flight*.

CCGBank

As with phrase-structure approaches, treebanks play an important role in CCG-based approaches to parsing. CCGBank (Hockenmaier and Steedman, 2007) is the largest and most widely used CCG treebank. It was created by automatically translating phrase-structure trees from the Penn Treebank via a rule-based approach. The method produced successful translations of over 99% of the trees in the Penn Treebank resulting in 48,934 sentences paired with CCG derivations. It also provides a lexicon of 44,000 words with over 1200 categories. Appendix C will discuss how these resources can be used to train CCG parsers.

12.7 Summary

This chapter has introduced a number of fundamental concepts in syntax through the use of **context-free grammars**.

- In many languages, groups of consecutive words act as a group or a **constituent**, which can be modeled by **context-free grammars** (which are 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 that can be **derived** from a particular **context-free grammar**.
- A **generative grammar** is a traditional name in linguistics for a formal language that 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 common types; these 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** and **infinitive** 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.
- **Treebanks** of parsed sentences exist for many genres of English and for many languages. Treebanks can be searched with 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.
- Lexicalized grammars place more emphasis on the structure of the lexicon, lessening the burden on pure phrase-structure rules.
- Combinatorial categorial grammar (CCG) is an important computationally relevant lexicalized approach.

Bibliographical and Historical Notes

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):

> *...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 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, the subject of Chapter 14.

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), the Minimalist Program (Chomsky, 1995), 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 reinvented 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. 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 Chapter 13).

Thre are various classes of extensions to CFGs, many designed to handle long-distance dependencies in the syntax. (Other grammars instead treat long-distance-dependent items as being related semantically rather than syntactically (Kay and Fillmore 1999, Culicover and Jackendoff 2005).

One extended formalism is **Tree Adjoining Grammar** (TAG) (Joshi, 1985). The primary TAG data structure 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, and auxiliary trees add recursion into a tree. Trees are combined by two operations called **substitution** and **adjunction**. The adjunction operation handles long-distance dependencies. See Joshi (1985) for more details. Tree Adjoining Grammar is a member of the family of **mildly context-sensitive languages**.

We mentioned on page 15 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).

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).

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 rules, unification, and the type hierarchy in Head-Driven Phrase Structure Grammar. Van Valin, Jr. and La Polla (1997) is an introduction from a **functional** perspective, focusing on cross-linguistic data and on the functional motivation for syntactic structures.

generative

functional

Exercises

12.1 Draw tree structures for the following ATIS phrases:

1. Dallas
2. from Denver
3. after five p.m.
4. arriving in Washington
5. early flights
6. all redeye flights
7. on Thursday
8. a one-way fare

9. any delays in Denver

12.2 Draw tree structures for the following ATIS sentences:

1. Does American Airlines have a flight between five a.m. and six a.m.?
2. I would like to fly on American Airlines.
3. Please repeat that.
4. Does American 487 have a first-class section?
5. I need to fly between Philadelphia and Atlanta.
6. What is the fare from Atlanta to Denver?
7. Is there an American Airlines flight from Philadelphia to Dallas?

12.3 Assume a grammar that has many *VP* rules for different subcategorizations, as expressed in Section 12.3.4, and differently subcategorized verb rules like *Verb-with-NP-complement*. How would the rule for postnominal relative clauses (12.4) 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*.

12.4 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**.

12.5 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

12.6 *NP*s like *Fortune's office* or *my uncle's marks* are called **possessive** or **genitive** noun phrases. We can model possessive noun phrases 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*).

12.7 Page 8 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.

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

Ajdukiewicz, K. (1935). Die syntaktische Konnexität. *Studia Philosophica 1*, 1–27. English translation "Syntactic Connexion" by H. Weber in McCall, S. (Ed.) 1967. *Polish Logic*, pp. 207–231, Oxford University Press.

Backus, J. W. (1959). The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference. *Information Processing: Proceedings of the International Conference on Information Processing, Paris*. UNESCO.

Backus, J. W. (1996). Transcript of question and answer session. Wexelblat, R. L. (Ed.), *History of Programming Languages*, p. 162. Academic Press.

Bar-Hillel, Y. (1953). A quasi-arithmetical notation for syntactic description. *Language 29*, 47–58.

Bazell, C. E. (1952/1966). The correspondence fallacy in structural linguistics. Hamp, E. P., Householder, F. W., and Austerlitz, R. (Eds.), *Studies by Members of the English Department, Istanbul University (3), reprinted in Readings in Linguistics II (1966)*, 271–298. University of Chicago Press.

Biber, D., Johansson, S., Leech, G., Conrad, S., and Finegan, E. (1999). *Longman Grammar of Spoken and Written English*. Pearson.

Bies, A., Ferguson, M., Katz, K., and MacIntyre, R. (1995). Bracketing guidelines for Treebank II style Penn Treebank Project..

Bloomfield, L. (1914). *An Introduction to the Study of Language*. Henry Holt and Company.

Bloomfield, L. (1933). *Language*. University of Chicago Press.

Bresnan, J. (Ed.). (1982). *The Mental Representation of Grammatical Relations*. MIT Press.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. *AAAI*.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory 2*(3), 113–124.

Chomsky, N. (1956/1975). *The Logical Structure of Linguistic Theory*. Plenum.

Chomsky, N. (1957). *Syntactic Structures*. Mouton, The Hague.

Chomsky, N. (1963). Formal properties of grammars. Luce, R. D., Bush, R., and Galanter, E. (Eds.), *Handbook of Mathematical Psychology*, Vol. 2, 323–418. Wiley.

Chomsky, N. (1995). *The Minimalist Program*. MIT Press.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia.

Culicover, P. W. and Jackendoff, R. (2005). *Simpler Syntax*. Oxford University Press.

Gazdar, G., Klein, E., Pullum, G. K., and Sag, I. A. (1985). *Generalized Phrase Structure Grammar*. Blackwell.

Harris, Z. S. (1946). From morpheme to utterance. *Language 22*(3), 161–183.

Hemphill, C. T., Godfrey, J., and Doddington, G. (1990). The ATIS spoken language systems pilot corpus. *Proceedings DARPA Speech and Natural Language Workshop*.

Hockenmaier, J. and Steedman, M. (2007). Ccgbank: a corpus of ccg derivations and dependency structures extracted from the penn treebank. *Computational Linguistics 33*(3), 355–396.

Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Huddleston, R. and Pullum, G. K. (2002). *The Cambridge Grammar of the English Language*. Cambridge University Press.

Joshi, A. K. (1985). Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions?. Dowty, D. R., Karttunen, L., and Zwicky, A. (Eds.), *Natural Language Parsing*, 206–250. Cambridge University Press.

Kay, P. and Fillmore, C. J. (1999). Grammatical constructions and linguistic generalizations: The What's X Doing Y? construction. *Language 75*(1), 1–33.

Magerman, D. M. (1995). Statistical decision-tree models for parsing. *ACL*.

Marcus, M. P., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. (1994). The Penn Treebank: Annotating predicate argument structure. *ARPA Human Language Technology Workshop*. Morgan Kaufmann.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics 19*(2), 313–330.

Naur, P., Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B., Wegstein, J. H., van Wijngaarden, A., and Woodger, M. (1960). Report on the algorithmic language ALGOL 60. *CACM 3*(5), 299–314. Revised in CACM 6:1, 1-17, 1963.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajič, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., and Zeman, D. (2016). Universal Dependencies v1: A multilingual treebank collection. *LREC*.

Percival, W. K. (1976). On the historical source of immediate constituent analysis. McCawley, J. D. (Ed.), *Syntax and Semantics Volume 7, Notes from the Linguistic Underground*, 229–242. Academic Press.

Pollard, C. and Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Pullum, G. K. (1991). *The Great Eskimo Vocabulary Hoax*. University of Chicago.

Quirk, R., Greenbaum, S., Leech, G., and Svartvik, J. (1985). *A Comprehensive Grammar of the English Language*. Longman.

Radford, A. (1997). *Syntactic Theory and the Structure of English: A Minimalist Approach*. Cambridge University Press.

Sag, I. A., Wasow, T., and Bender, E. M. (Eds.). (2003). *Syntactic Theory: A Formal Introduction*. CSLI Publications, Stanford, CA.

Steedman, M. (1989). Constituency and coordination in a combinatory grammar. Baltin, M. R. and Kroch, A. S. (Eds.), *Alternative Conceptions of Phrase Structure*, 201–231. University of Chicago.

Steedman, M. (1996). *Surface Structure and Interpretation*. MIT Press. Linguistic Inquiry Monograph, 30.

Steedman, M. (2000). *The Syntactic Process*. The MIT Press.

Van Valin, Jr., R. D. and La Polla, R. (1997). *Syntax: Structure, Meaning, and Function*. Cambridge University Press.

Wundt, W. (1900). *Völkerpsychologie: eine Untersuchung der Entwicklungsgesetze von Sprache, Mythus, und Sitte*. W. Engelmann, Leipzig. Band II: Die Sprache, Zweiter Teil.