The Structure of the Lexicon

8.1 Introduction

In the course of the last few chapters, we have put more and more of the descriptive burden of our theory into the lexicon. Lexical entries have evolved from simple pairings of phonological forms with grammatical categories into elaborate information structures, in which phonological forms are now paired with more articulated feature structure descriptions. This has permitted us to reduce our inventory of grammar rules to a few very general schemas, and to account for a range of syntactic phenomena.

Since our theory relies heavily on rich lexical representations, we need to consider what kind of internal organization the lexicon should have. In particular, we do not want to claim that all information contained in lexical entries is simply listed. A great number of the constraints that we are now putting into lexical entries are not idiosyncratic to individual words. Rather, they reflect general properties of classes of words, e.g. common nouns, proper nouns, verbs, tensed verbs, and so forth. Stipulating all of these constraints redundantly on each individual lexical item would simply miss all the significant generalizations about how words and lexical constraints are organized. For example, we handle subject-verb agreement by having the AGR value of a verb be the same as the AGR value of its specifier. We guarantee that this identity holds by imposing the SHAC on a lexical class that includes verbs. Most verbs have two lexical entries that are present tense, one whose AGR value is of type \textit{3sing} and another whose AGR value is \textit{non-3sing}. Aside from the difference in their AGR value (and hence of their specifiers’ AGR values), these two entries for each verb are essentially identical: their part of speech is \textit{verb}; they have the same COMPS value; and their semantics includes the same predication. This is no accident, nor is the fact that the same suffix (namely, -\text{s}) is used to mark almost all third-person singular present tense verb forms.

Notice, by the way, that capturing such generalizations is motivated not only by general considerations of parsimony, but also on psychological grounds. On encountering a novel English verb (say, a recent coinage such as \textit{email} or an obscure word like \textit{cark}), any competent speaker will add the suffix -\text{s} when using it in the present tense with a third person singular subject. In short, speakers know that there are systematic (or, as linguists say, ‘productive’) relationships among different forms of the same word, and our grammar should reflect this systematicity. The focus of the present chapter is to develop mechanisms for expressing regularities within the lexicon.
8.2 Lexemes

Before we begin developing our lexical theory, however, we want to call attention to what, in everyday English, are two different uses of the term ‘word’. In some contexts, people informally distinguish, for example, _runs_ and _run_ as two different words; they are pronounced differently, have (subtly) different meanings, and have slightly different co-occurrence restrictions. But in other contexts, the same people would have no hesitation in referring to _runs_ and _run_ as two forms of the word _run_. Clearly, these are two different conceptions of ‘word’: the first refers to a certain pairing of sound and meaning, whereas the latter refers to a family of such pairings. In a formal theory of grammar, these two concepts must not be conflated. Our type _word_ corresponds to the first usage (in which _runs_ and _run_ are distinct words). The lexical entries that give rise to word structures must all be of type _word_.

But we also want to capture what people have in mind when they use ‘word’ in the second sense. That is, we want to be able to express the relationship between _runs_ and _run_ (and _run_ and _running_). We do this by means of a new type _lexeme_. A _lexeme_ can be thought of as an abstract proto-word, which, by means to be discussed in this chapter, gives rise to genuine words (that is, instances of the type _word_).

Note that in any language with a rich system of morphological inflection, the need for the notion of ‘lexeme’ would be apparent. In Spanish, for example, we find paradigms of related words like the following:

<table>
<thead>
<tr>
<th>(1)</th>
<th>vivo</th>
<th>‘I live’</th>
<th>vives</th>
<th>‘you(sg.) live’</th>
<th>vive</th>
<th>‘(s)he/it lives’</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>vivimos</td>
<td>‘we live’</td>
<td>vivís</td>
<td>‘you(pl.) live’</td>
<td>viven</td>
<td>‘they live’</td>
</tr>
<tr>
<td></td>
<td>vivía</td>
<td>‘I lived’</td>
<td>vivías</td>
<td>‘you(sg.) lived’</td>
<td>vivía</td>
<td>‘(s)he/it lived’</td>
</tr>
<tr>
<td></td>
<td>vivíamos</td>
<td>‘we lived’</td>
<td>vivíais</td>
<td>‘you(pl.) lived’</td>
<td>vivían</td>
<td>‘they lived’</td>
</tr>
<tr>
<td></td>
<td>vivire</td>
<td>‘I’ll live’</td>
<td>vivirás</td>
<td>‘you(sg.)’ll live’</td>
<td>vivira</td>
<td>‘(s)he/it’ll live’</td>
</tr>
<tr>
<td></td>
<td>viviremos</td>
<td>‘we’ll live’</td>
<td>vivireis</td>
<td>‘you(pl.)’ll live’</td>
<td>vivirán</td>
<td>‘they’ll live’</td>
</tr>
</tbody>
</table>

Clearly we need some way of talking about what these forms all have in common. We will say that they are distinct words associated with – or derived from – a common lexeme. Each such lexeme contributes a unique constellation of information – partly phonological (the stem from which all these inflected forms are derived), partly syntactic (including, among other things, the information that this is a verbal lexeme), partly semantic (the meaning that distinguishes this from other verbal lexemes). The reason why it isn’t so obvious that we need a notion like lexeme in English is simply that English (for historical reasons) has very little inflectional morphology. Nonetheless, we’ll be happy to have a way of analyzing a family of forms like the following, all of which are realizations of a common lexeme:

| (2) | do, does, did, don’t, doesn’t, didn’t, doing |

We incorporate the notion of lexeme into our theory by first revising a high-level distinction in our type hierarchy – the types that distinguish among the syntactic-semantic complexes we have been referring to as expressions, words, and phrases. We will refer to the most general such type of feature structure simply as _synsem_ (indicating that it is a complex of syntactic and semantic information). The type _expression_ will then be
an immediate subtype of synsem, as will the new type lexeme. And, as before, word and phrase are the two immediate subtypes of expression. This reorganization of the type hierarchy is summarized in (3):

(3)  
\[ \text{feat-struc} \]
\[ \text{synsem} \]
\[ \text{expression} \]
\[ \text{lexeme} \]
\[ \text{word} \]
\[ \text{phrase} \]

The feature ARG-ST is defined for both lexeme and word, so both lexemes and words have argument structure.¹

Up to now, we have simply stated most lexical constraints in individual lexical entries. For example, whatever generalizations hold for all common nouns have been stipulated redundantly in each common noun’s lexical entry. The same is true for the lexical entries we have posited for verbs. But there are many regularities that hold over classes of lexemes – common noun, proper noun, intransitive verb, transitive verb, and so forth. We will now modify our grammar in order to be able to express these generalizations.

Just as we have used a type hierarchy to factor out general properties of linguistic objects in terms of type constraints, our grammar will now organize lexemes into subtypes of the type lexeme, in order to provide a home for generalizations about word classes. We’ll deal with regularities governing inflectional classes (third-singular present tense verbs, plural nouns, etc.) in terms of Lexical Rules, a new construct we introduce and explain in section 8.6 below.

### 8.3 Default Constraint Inheritance

In previous chapters, we introduced the idea that some types are subtypes of others, with the following effect:

(4) If T₂ is a subtype of T₁, then
   a. every feature specified as appropriate for T₁ is also appropriate for T₂, and
   b. every constraint associated with T₁ affects all instances of T₂.

Formulated in this way, the inheritance of constraints in our type hierarchy is MONOTONIC: constraints on supertypes affect all instances of subtypes, without exception. An intuitive alternative to this conception is to allow for DEFEASIBLE constraints – constraints on a given type that hold BY DEFAULT, i.e. unless contradicted by some other constraint that holds at a more specific level. In this alternative picture, contradictory information associated with a subtype takes precedence over (or overrides) defeasible constraints that would otherwise be inherited from a supertype. Defeasible constraints and default inheritance allow a type system to express the idea that language embodies generalizations that have exceptions – subclasses with subregularities and individual

¹Strictly speaking, a grammar cannot allow a given feature to be specified for two incompatible types. Hence, the current hierarchy, where lexeme and word are incompatible, is defective. In Chapter 16 we remedy this defect by recasting the lexicon as a multiple inheritance hierarchy.
elements with idiosyncratic properties.

It has long been recognized that the default generalizations we find in natural languages are layered, i.e., that there are default generalizations governing intermediate-level categories of varying grain. This intuitive idea is simple to express: we need to allow a constraint associated with a given lexical type to be marked as defeasible. Suppose a defeasible constraint $C_i$ applies to a lexical type $T_i$. Then this constraint holds of any lexical entry of type $T_i$ for which it is not explicitly contradicted. It could be overridden in one of two ways. First, a subtype of $T_i$ might have a constraint associated with it that contradicts $C_i$. That is, there could be a type $T_j$ that is a subtype of $T_i$ and a constraint $C_j$ associated with $T_j$ that is incompatible with $C_i$:

\[
\begin{array}{ccc}
T_i & : & C_i \\
\downarrow & & \downarrow \\
\ldots & & \ldots \\
\downarrow & & \downarrow \\
T_j & : & C_j \\
\ldots & & \ldots \\
\downarrow & & \downarrow \\
T_m & & \ldots
\end{array}
\]

In this case, $C_j$ takes precedence and overrides $C_i$. A second way to override a defeasible constraint involves information stipulated in a particular lexical entry. That is, a constraint on a particular instance of a leaf type $T_m$ (a subtype of $T_i$) could contradict $C_i$.\(^2\) In this case, too, the information associated with the lexical entry takes precedence over the defeasible constraint. But that constraint is true of all instances of $T_i$ in which it is not overridden (as of course are all nondefeasible constraints).

Natural languages exhibit a great many regularities with exceptions that are modeled elegantly in terms of type hierarchies. For example, names in English (often called proper nouns) don't usually take specifiers. This is illustrated in (6):

(6) a. Cameron skates.
b. \{"A", "The"\} Cameron skates.

Moreover, proper nouns are normally third-person singular, as (7) shows:

(7) *Cameron skate.

These generalizations will be captured in our type system by introducing a type for proper nouns with defeasible constraints (stated more formally below) specifying that the value of AGR must be of type $3sing$ and that the ARG-ST (and hence both SPR and COMPS lists) must be empty. But there are exceptions to these constraints. In particular, there

---

\(^2\)Recall that a leaf type (also known as a 'maximal' type) is a type that has no subtypes. We're taking a small liberty here in talking of the lexical entry as describing an instance of a leaf type. In our current set-up (but not the one discussed in Chapter 16), our lexical entries in fact describe pairs consisting of a form and a feature structure belonging to a leaf-type. We will sometimes say, informally, that a lexical entry is of some particular type. What we mean by this is that the second element of (the ordered pair that makes up) the lexical entry describes feature structures of that type.
are several proper nouns in English naming mountain ranges that appear only in the plural and only with a determiner:

(8) a. The \{Andes, Alps\} are magnificent.

b. *The \{Ande, Alp\} is magnificent.

c. Hannibal crossed the \{Alps, Andes\}

d. *Hannibal crossed \{Alps, Andes\}

In fact, names for mountain ranges may be a lexical type in the lexeme hierarchy, providing an example of a lexical subtype whose constraints override two constraints on a superordinate type.

An even clearer example of this phenomenon is names for US sports teams. In every team sport in the United States, it is in general true that the team names are plural and select the as their specifier:

(9) a. The San Francisco Giants \{are\} \{is\} in first place.

b. The (Bay Area) CyberRays \{were\} \{was\} in Boston yesterday.

c. The (Oakland) Raiders \{play\} \{plays\} in Denver tonight.

d. The (Boston) Celtics \{play\} \{plays\} Indiana today.

An alternative hypothesis about the names of mountain ranges and team names is to treat them as ‘words with spaces in them’, including the as part of the proper noun’s form. Such an analysis would treat these names as having the same SPR value (\{\}) as all other proper nouns. The ‘words with spaces’ analysis is presumably necessary for other names, e.g. San Francisco, Great Britain, or \{The\} Leland Stanford Junior University Marching Band. However, there is evidence that the proper nouns Andes, Oakland Raiders, or Boston Celtics (unlike San Francisco and the like) must be entered in the lexicon as nouns that combine with a specifier syntactically because of other regularities having to do with compound nouns.

Compound nouns can be constructed from pairs of nouns:

(10) a. car thief

b. department chair

c. community center
As (10) shows, the first member of the compound can be either a common noun or a proper noun. And these compound nouns, once constructed (by a lexical rule), can combine syntactically with a determiner in the same way that a non-compound common noun does:

(11) a. \( \text{a the [car thief]} \)
    b. \( \text{a the [department chair]} \)
    c. \( \text{a the [community center]} \)
    d. \( \text{a the [Boston lawyer]} \)
    e. \( \text{an the [Oakland mayor]} \)

By including *Andes*, *Oakland Raiders*, and *Boston Celtics* in the lexicon as nouns that select for a determiner syntactically (rather than listing *the Andes*, *the Oakland Raiders* and *the Boston Celtics*), we correctly predict their behavior in compound nouns. That is, it is the determinerless elements that form compounds with other nouns:

(12) a. \( \text{an the [Andes specialist]} \)
    b. \( \text{an the [([Oakland) Raiders] spokesperson]} \)
    c. \( \text{a the [([Boston) Celtics] player]} \)

If we were to treat names for mountain ranges and sports teams as ‘words with spaces in them’, we would incorrectly predict that compound nouns like the following would be well-formed:

(13) a. \( \text{a the [the Andes] specialist] \)
    b. \( \text{a the [the Oakland Raiders] spokesperson]} \)
    c. \( \text{a the [the Boston Celtics] manager]} \)

Hence there is independent justification for our claim that these classes of proper noun are exceptional both in being plural and in selecting a specifier.
Note further that there are exceptions to the subregularity of sports team names. Certain US teams have names that are combinations of determiner plus mass noun:

(14)  a. The (Miami) Heat  
      b. The (Philadelphia) Charge  
      c. The (Stanford) Cardinal\(^3\)

These determiner-selecting nouns have singular uses, as the following examples show (though there appears to be some variation in this):

(15)  a. Despite their average age, the Charge boasts an experienced roster.\(^4\)  
      b. The Cardinal plays Arizona State at 7 p.m Saturday at Stanford.\(^5\)

This is a typical situation: many broad and productive generalizations in languages have exceptions, either idiosyncratic lexical items or classes of idiosyncratic expressions. For this reason, we shall adopt the method of default inheritance of (defeasible) constraints in our type hierarchy. This will allow us both to restrict the number of types that are required in our grammar and also to keep our constraints simple, without precluding the possibility that some instances or subtypes might be exceptions to the constraints.

By organizing the lexicon as a type hierarchy, together with the use of default inheritance, as described above, we can minimize the stipulations associated with particular lexical entries and express the shared properties of different word classes, at the same time that we allow for idiosyncrasy of the kind we have been discussing. The overall conception of the lexicon is as shown in (16):

(16)  \[
\text{lexeme} \quad \rightarrow \quad \ldots \quad \rightarrow \quad \ldots \\
\quad \rightarrow \quad T_i \quad \rightarrow \\
\quad \rightarrow \quad \ldots \quad \rightarrow \\
\quad \rightarrow \quad T_m \quad \rightarrow \\
\]

Each of our basic lexical entries will include a feature structure assigned to some maximal (that is, leaf) type \(T_m\). \(T_m\) will in turn have a family of supertypes \(T_i\), that are intermediate between the type \textit{lexeme} and \(T_m\). The various intermediate types correspond to intermediate levels of classification, where type constraints can express linguistic generalizations. Each type in the lexeme hierarchy (which elaborates the hierarchy shown earlier in (3)) has constraints associated with it — some \textit{inviolable}, and others that are defeasible. Since this is a default inheritance hierarchy, we can provide a natural account of the fact that individual lexemes have many properties in common but may differ from one another in terms of particular constraints that override the general constraints governing their supertypes. The idea is that each (basic) lexical item describes a distinct family of lexemes, each of which is an instance of a maximal type \(T_m\). The members

---

\(^3\)This name refers to the color, not the bird.  
of that family inherit the constraints stipulated with the given lexical entry, the constraints associated with \( T_m \), and those associated with the supertypes of \( T_m \). A lexeme inherits the inviolable constraints and all compatible default constraints. Once a lexical hierarchy (with associated constraints) is put into place, the lexical entries that we write become highly streamlined (perhaps indicating no more than its phonology and meaning and which maximal type its satisfiers belong to). All further grammatically relevant constraints (i.e. the rest of the constraints that need to be included in the lexical entry) are inherited automatically, according to the method just described.

We use the symbol `/` to indicate that a certain specification is defeasible and hence can be overridden by a conflicting specification.\(^6\) As a simple example of a defeasible constraint, let us go back to the framework for modeling universities we presented in Chapter 3. Suppose we wanted to adapt the system presented there to model New College, a college so small that it relies almost exclusively on a single telephone number. If only the most important individuals had their own telephone number, we might hypothesize a defeasible constraint like the following:

\[
\text{(17)} \quad \text{entity, } \left[ \text{TEL / 555-111-1234} \right] \]

Our entry for the New College Music Department (analogous to one of our ‘lexical entries’) might then be as shown in (18):

\[
\text{(18)} \quad \left[ \text{department} \right. \\
\text{NAME} & \quad \text{New College Music} \\
\text{FOUNDERS} & \quad \left[ \text{NAME LaVern Baker} \right. , \left[ \text{NAME Clyde McPhatter} \right. \\
\text{CHAIR} & \quad \left[ \text{NAME Johnny Otis} \right. \left. \right] \right. \\
\left. \right] \\
\text{TEL} & \quad 555-111-1234 \\
\left. \right] \\
\]

Because \text{department} is a subtype of \text{entity}, all instances of the type \text{department} inherit the constraint in (17), unless their entry says otherwise. Thus New College Music has the properties shown in (19), but New College English could have an entry like (20), which overrides (17):

\[
\text{(19)} \quad \left[ \text{department} \right. \\
\text{NAME} & \quad \text{New College Music} \\
\text{FOUNDERS} & \quad \left[ \text{NAME LaVern Baker} \right. , \left[ \text{NAME Clyde McPhatter} \right. \\
\text{CHAIR} & \quad \left[ \text{NAME Johnny Otis} \right. \left. \right] \right. \\
\text{T\textsc{el}} & \quad 555-111-1234 \\
\left. \right] \\
\]

\(^6\)The theory of defaults we employ here (as well as the `/` notation) is adapted from Lascarides et al. (1996). See also Lascarides and Copestake (1999) and the further readings at the end of this chapter.
We will also sometimes want to indicate that two feature values are identical by default. We can also do this using the `/` notation. In Chapter 3, we considered a constraint requiring that a department and its head have the same telephone number. As we noted there in passing, this constraint is not true of Stanford. But suppose it were the norm, with only occasional exceptions. In that case, we could include in our theory a defeasible version of that constraint, which would be formulated as follows:

(20) \[
\begin{array}{l}
\text{department} \\
\text{Name} \quad \text{New College English} \\
\text{Founders} \quad \{\ \} \\
\text{Chair} \quad \{\text{Name} \quad \text{Lawrence Ferlinghetti}\} \\
\text{Tel} \quad 555-111-5555
\end{array}
\]  

This constraint allows an individual department chair to have a phone number distinct from that of the department [s]he chairs, but will enforce the relevant identity, unless there is some specific indication to the contrary. A similar constraint might indicate that the chair of a New College department is its founder, by default. Defeasible identity constraints are a bit tricky, though – we will consider them in more detail in section 8.6 below.

There is one final property of our approach to default constraint inheritance that is important to understand. This has to do with the behavior of complex defeasible constraints. Suppose some type in our grammar \( T_i \) requires that the value of the feature MOD be \( \{ S \} \), by default. Given that ‘S’ is an abbreviation, this constraint could be formulated more precisely as in (22):

\[
T_i : \left[ \begin{array}{l}
\text{Syn} \\
\text{Val} \\
\text{Mod} \left( \left[ \begin{array}{l}
\text{Head} \\
\text{Verb}
\end{array} \right] \right) \right]
\]  

Here the default specification involves three features: HEAD, SPR and COMPS.

Suppose now that \( T_j \), a subtype of \( T_i \), contradicts just part of the constraint in (22), say as in (23):

\[
T_j : \left[ \begin{array}{l}
\text{Syn} \\
\text{Val} \\
\text{Mod} \left( \left[ \begin{array}{l}
\text{Syn} \\
\text{Val} \\
\text{Spr} \left( \{ \} \right) \\
\text{Comps} \left( \{ \} \right)
\end{array} \right] \right) \right]
\]  

The important thing to understand about the interaction of complex defaults like (22) and constraints like (23) is that defeasible constraints that are not explicitly contradicted remain in force. That is, the combination of (22) and (23) is the constraint shown in (24), where only the information specifically contradicted is overridden:
Note that the default part of the constraint has been ‘pushed down’ to the next level of embedding in such a way as to have the maximum effect that is still consistent with the overriding constraint. Instances of type $T_i$ are thus S-modifiers by default, but instances of the subtype $T_j$ are VP-modifiers.7 ‘Mixed’ constraints like (24) are sometimes written as shown in (25), where the ‘hard’ constraints appear before the ‘/’ and the defeasible parts appear afterwards:

\[
T_j : \left[ \begin{array}{c}
\text{SYN} \\
\text{VAL} \\
\text{MOD} \\
\text{SYN} \\
\text{VAL} \\
\text{SPR} \\
\text{COMPS} \\
\end{array} \right] \left[ \begin{array}{c}
\text{HEAD} \\
\text{verb} \\
\end{array} \right] \left[ \begin{array}{c}
\text{SPR} \\
\text{NP} \\
\end{array} \right] \left[ \begin{array}{c}
\text{COMPS} \\
\end{array} \right] \left[ \begin{array}{c}
\text{verb} \\
\end{array} \right]
\]

8.4 Some Lexemes of Our Grammar

The basic lexical entries, taken together with the constraints inherited via the lexeme hierarchy, characterize the set of basic lexical elements of the language. These are one kind of LEXICAL SEQUENCE, pairs consisting of a phonological form and a feature structure of type lexeme.8 Each such lexical sequence then gives rise to a family of another kind of lexical sequence whose second member is a word, more precisely a feature structure of type word. This is accomplished through the application of inflectional rules. Thus, lexical entries serve as the material that gets shaped into words, and words serve as the building blocks for syntactic structures. In the next section (and much of the remainder of this book) we will discuss lexical rules.9

Many of the constraints we present here are stated solely in terms of ARG-ST specifications. These constraints affect lexemes, and hence lexical entries, directly, but they also indirectly affect words derived from those entries. Together with the ARP (developed in the last chapter), these constraints on ARG-ST are reflected in the SPR and COMPS values of the relevant word structures.10

---

7The constraints on the HEAD and COMPS values in (24) are defeasible because the constraints on $T_j$ may still be overridden by constraints on one of its subtypes or by constraints on a particular lexical entry.

8In Chapter 16, we will show how lexical sequences can be eliminated, once the notion ‘sign’ is introduced.

9We are assuming that even noninflected words are derived from lexemes. An alternative that we will not pursue here is to enter such words directly into the lexicon with no corresponding lexemes.

10Note that the value of ARG-ST, as before, is a list of feature structures of type expression. This now has the important effect of disallowing lexemes as members of ARG-ST lists. Since ARG-ST elements correspond to members of SPR and COMPS lists, and these correspond to the elements selected by the heads of phrases (i.e. to the non-head daughters in our headed phrases), the fact that arguments must be expressions also entails that lexemes cannot appear as specifiers or complements in our syntactic structures. In fact, we want all daughters in syntactic structures to be expressions, rather than lexemes, and will make further modifications in our grammar rules to ensure this.
As noted earlier, we are now assuming that lexeme and expression are the two immediate subtypes of the type synsem and that word and phrase are the two immediate subtypes of expression. The type lexeme bears the constraints in (26):

\[
\text{lexeme : } \begin{bmatrix} \text{ARG-ST} & \text{list(expression)} \\ \text{SYN} & \text{VAL} \begin{bmatrix} \text{MOD} & /() \end{bmatrix} \end{bmatrix}
\]

These constraints declare the feature ARG-ST to be appropriate for all lexemes, and make [MOD /e()] the default, as most lexemes cannot be modifiers.

Among lexemes, we draw a further distinction between those that give rise to a set of inflected forms and those that do not show any morphological inflection. That is, we posit inflecting-lexeme (inf/-lxm) and constant-lexeme (const/-lxm) as two subtypes of lexeme. The type hierarchy we will assume for nominal and verbal lexemes in English is sketched in (27):

Here, each leaf type corresponds to a lexical class and the various supertypes correspond to larger classes that exhibit regularities that are shared by more than one of the smallest classes. We will explain each of these types in turn.

Before turning to the relevant leaf types, however, we need to comment briefly on three types at the top of the lexeme hierarchy. Inflecting lexemes are further classified in terms of the subtypes common-noun-lexeme (cn-lxm) and verb-lexeme (verb-lxm), as these are the only two kinds of English lexeme considered here that give rise to inflected forms. The types proper-noun-lexeme (pn-lxm) and pronoun-lexeme (pron-lxm) are two of the subtypes of const-lxm. They are discussed more fully in section 8.4.3 below. This organization has the benefit of providing a natural home for the SHAC. It is now a constraint on the type inf/-lxm.
Specifier-Head Agreement Constraint (SHAC):

\[
\begin{bmatrix}
\text{SPECIFIER} & \text{HEAD} \\
\text{SPECIFIER-} & \text{AGR} [\text{PER 3rd}] \\
\text{INF-LXM} & \text{VAL} [\text{SPR} \langle [\text{AGR} [\text{PER 3rd}]]\rangle] \\
\text{SPECIFIER} & \text{ARG-ST} \langle X \oplus [\text{ref}] \rangle \\
\text{INF-LXM} & \text{SEM} \langle \text{MODE} [\text{ref}] \rangle
\end{bmatrix}
\]

The SHAC has two effects: it ensures that elements select for a specifier and that they agree with the specifiers they select. As desired, the SHAC applies only to verbs and to common nouns. Notice that the SHAC is not a defeasible constraint.

8.4.1 Nominal Lexemes

The type cn-lxm exhibits numerous regularities that are summarized by the complex constraint in (29):

\[
\begin{bmatrix}
\text{SPECIFIER} & \text{HEAD} \\
\text{SPECIFIER-} & \text{AGR} [\text{noun}] \\
\text{INF-LXM} & \text{VAL} [\text{SPR} \langle [\text{HEAD det}] \rangle] \\
\text{SPECIFIER} & \text{ARG-ST} \langle X \oplus [\text{ref}] \rangle \\
\text{INF-LXM} & \text{SEM} \langle \text{MODE} [\text{ref}] \rangle
\end{bmatrix}
\]

(29) ensures that all common nouns are [HEAD noun], that they select determiners as their specifier, and that the rest of their ARG-ST is the empty list, by default. Since common nouns – words derived from common noun lexemes – must select for a specifier argument,\textsuperscript{11} this will mean, once we factor in the effect of the ARP, that their COMPS list is empty, by default. A noun like picture, which takes an optional PP complement in examples like (30), provides part of the motivation for making this specification defeasible:

\[
\begin{align*}
(30) & \quad \text{a. } \text{[The (picture (of Sandy))] was awesome.} \\
& \quad \text{b. } \text{We couldn’t find [any (pictures (of Blind Mello Jello))].}
\end{align*}
\]

Finally, note that (29) also requires that common nouns be referential ([MODE ref]), by default. This is a defeasible constraint because in Chapter 11 we will encounter some common nouns that are not referential.

The type cn-lxm has two subtypes: count-noun-lxm (cntn-lxm) and mass-noun-lxm (massn-lxm). These are constrained as shown in (31):

\[
\begin{align*}
(31) & \quad \text{a. } \text{cntn-lxm : } \begin{bmatrix}
\text{SPECIFIER} & \text{VAL} \\
\text{SPECIFIER-} & \text{SPR} \langle [\text{COUNT } +] \rangle \\
\text{INF-LXM} & \text{SEM} \langle \text{MODE} [\text{ref}] \rangle
\end{bmatrix} \\
& \quad \text{b. } \text{massn-lxm : } \begin{bmatrix}
\text{SPECIFIER} & \text{VAL} \\
\text{SPECIFIER-} & \text{SPR} \langle [\text{COUNT } -] \rangle \\
\text{INF-LXM} & \text{SEM} \langle \text{MODE} [\text{ref}] \rangle
\end{bmatrix}
\end{align*}
\]

\textsuperscript{11} The claim that specifiers are obligatory for common nouns appears to be inconsistent with the existence of plural and mass NPs that lack determiners. The analysis of such NPs is the topic of Problem 2 below.
These type constraints allow the lexical entries for common nouns to be very streamlined indeed. (32) is a typical lexical entry for a count noun in our grammar:

\[
\langle \text{dog} , \\
\text{cntn-\text{lxm}} \\
\text{SEM} \\
\begin{array}{l}
\text{INDEX} i \\
\text{RESTR} \left\langle \text{RELN \text{dog}} \right\rangle \\
\text{INST} i
\end{array}
\rangle
\]

Here, as before, the second member of this lexical entry is a feature structure description.

What objects satisfy an entry like (32)? Here again (as in the case of feature structures in word structures - see Chapter 6, Section 6.2.1), the second element in (32) can correspond to infinitely many resolved feature structures. Hence there are infinitely many lexical sequences that satisfy a lexical entry like (32). These lexical sequences are the ones that satisfy the constraints stated in (31) and (32) as well as all of the constraints inherited from the supertypes of \text{cntn-\text{lxm}}. We represent the family of such lexical sequences as in (33), where we show all of the constraints inherited by the feature structure in the pair. The shading of the type name indicates that (33) is a concise representation of a family of lexical sequences, each of whose resolved feature structure contains more information than what is shown.

\[
\langle \text{dog} , \\
\text{cntn-\text{lxm}} \\
\text{SYN} \\
\begin{array}{l}
\text{HEAD} \\
\text{AGR} \left\langle \text{PER \text{3rd}} \right\rangle \\
\text{VAL} \\
\text{SPR} \left\langle \text{det} \right. \\
\text{AGR} \left. \left\langle \text{COUNT} + \right\rangle \right\rangle \\
\text{MODE} \text{ref} \\
\text{INDEX} i \\
\text{RESTR} \left\langle \text{RELN \text{dog}} \right\rangle \\
\text{INST} i
\end{array}
\rangle
\]

We may take this family of feature structures to be the lexeme \text{dog}. For reasons discussed above, however, none of these feature structures can yet enter into a grammatical structure. The role of a lexeme, described more fully in the next section, is to license the existence of a family of words, and it is these words that are used to construct larger structures.

Given the type hierarchy and constraints just outlined, the rather complex set of specifications that we want to associate with a particular lexeme can be largely predicted simply by associating the lexeme with the appropriate type in the lexeme hierarchy.
Essentially, all that remains to be stipulated in a given lexical entry is its phonological form, the particular predication in its semantic restriction, and any exceptional properties it may have. The rest follows from ‘the logic of the lexicon’. This is precisely what lexical stipulation should be reduced to, whenever possible.

Exercise 1: Understanding Constraint Inheritance

You should make sure you understand why (33) contains exactly the information it does. For each constraint in (33), identify which type it is a constraint on. In addition, identify all cases of overriding of defeasible constraints.

Proper nouns and pronouns instantiate the types \textit{pn-lxm} and \textit{pron-lxm}, which are constrained as follows:

\begin{equation}
\begin{aligned}
(34) \quad & a. \\
& \text{\textit{pn-lxm}:} \\
& \text{SYN} \\
& \text{HEAD} \\
& \text{AGR} \\
& \text{PER} \\
& \text{NUM} \\
& \text{SEM} \\
& \text{MODE ref} \\
& \text{ARG-ST / } \langle \rangle \\
\end{aligned}
\end{equation}

\begin{equation}
\begin{aligned}
& b. \\
& \text{\textit{pron-lxm}:} \\
& \text{SYN} \\
& \text{HEAD noun} \\
& \text{SEM} \\
& \text{MODE / ref} \\
& \text{ARG-ST / } \langle \rangle \\
\end{aligned}
\end{equation}

These constraints require all proper nouns and pronouns to be [\text{HEAD noun}]. It also ensures that proper nouns are referential and that, by default, they are singular and have an empty ARG-ST list. As we saw at the beginning of this chapter, there are systematic exceptions to these last two constraints. (34b), on the other hand, imposes the nondefeasible constraint that pronouns have an empty ARG-ST list. There are no exceptional pronouns analogous to the names of mountain ranges or US sports teams. We have already seen pronouns whose MODE value is ‘ana’, rather than ‘ref’. In addition, in Chapter 11 we will see examples of nonreferential pronouns. For both these reasons, the referentiality requirement in (34b) is defeasible, as indicated.

8.4.2 Verbal Lexemes

The next class of lexemes to consider is verbs. As we saw in Chapter 3, all verbs have certain properties in common, but there are also subclasses of verbs that differ from one another in systematic ways. Until now, we’ve had to stipulate these differences for each and every verb. In this section, we will see how the type hierarchy can capture generalizations about those subclasses.

Because \textit{verb-lxm} is a subtype of \textit{inf-lxm}, the SHAC guarantees that verbal lexemes will select for an agreeing specifier. In addition to this inherited constraint, we require that any instance of the type \textit{verb-lxm} must have a HEAD value of type \textit{verb} and the
The Structure of the Lexicon

MODE value ‘prop’. In addition, the argument structure of a lexeme of this type begins with an NP. (In Chapter 11, we discuss verbs that take non-NP subjects. This will lead us to revise this constraint.) The constraints just noted are consolidated into (35):\(^\text{12}\)

\[
\text{verb-}lxm \quad \begin{bmatrix}
\text{SYN} \\
\text{SEM} \\
\text{ARG-ST}
\end{bmatrix}
\begin{bmatrix}
\text{HEAD verb} \\
\text{MODE prop} \\
\langle \text{NP, ...} \rangle
\end{bmatrix}
\]

As we saw earlier, the various subtypes of \text{verb-}lxm as distinguished by their ARG-ST specifications. The relevant part of our lexeme hierarchy is repeated in (36):

\[
\text{verb-}lxm \quad \begin{bmatrix}
\text{arg-st}
\end{bmatrix}
\begin{bmatrix}
\text{siv-}lxm \\
piv-\text{lxm} \\
tv-\text{lxm}
\end{bmatrix}
\begin{bmatrix}
\text{arg-st}
\end{bmatrix}
\begin{bmatrix}
\langle X \rangle \\
\langle X, PP \rangle \\
\langle X, NP, ... \rangle
\end{bmatrix}
\]

\[
\begin{bmatrix}
\text{arg-st}
\end{bmatrix}
\begin{bmatrix}
\text{stv-}lxm \\
dtv-\text{lxm} \\
piv-\text{lxm}
\end{bmatrix}
\begin{bmatrix}
\text{arg-st}
\end{bmatrix}
\begin{bmatrix}
\langle X, Y \rangle \\
\langle X, Y, NP \rangle \\
\langle X, Y, PP \rangle
\end{bmatrix}
\]

Here we have introduced the type \textit{transitive-verb-lexeme} (\textit{tv-}lxm) as a sister of the two intransitive verb types \textit{strict-intransitive-verb-lexeme} (siv-\textit{lxm}) and \textit{prepositional-intransitive-verb-lexeme} (piv-\textit{lxm}). Instances of siv-\textit{lxm} take no complements at all (e.g. \textit{sleep}); instances of piv-\textit{lxm} take a PP complement (e.g. \textit{rely}).\(^\text{13}\)

(37) a. Leslie slept (*the baby).

b. Dana relied *(on Hilary).

\(^{12}\text{Note that if lists are represented in terms of the features FIRST and REST (as they often are in computational work), then the proper formulation of the defeasible ARG-ST constraint in (34) would be as in (i):}\)

\[
(i) \begin{bmatrix}
\text{ARG-ST}
\end{bmatrix}
\begin{bmatrix}
\text{FIRST / NP}
\end{bmatrix}
\]

\(^{13}\text{We use the notation of an asterisk outside of the parentheses to mean that the sentence is ungrammatical without the parenthetical material. An asterisk inside the parentheses means the sentence is ungrammatical with the parenthetical material.}\)
Similarly, the transitive verb lexemes are subclassified into *strict-transitive-verb-lexeme* (stv-lxm, e.g. *devour*), *ditransitive-verb-lexeme* (dtv-lxm, e.g. *hand*), and *prepositional-transitive-verb-lexeme* (ptv-lxm, e.g. *put*):

(38) a. Pat devoured *(the sandwich).*
    b. Chris handed *(Bo) *(a ticket).*
    c. We put *(the book) *(on the shelf).*

As before, these types and their associated constraints (shown in (36)) allow us to replace lexical stipulation with type-based inference.

Thus, by adding a lexical entry like (39), we ensure that there is a family of lexical sequences like (40):

(39) \[
\begin{array}{c}
\text{dtv-lxm} \\
\langle \text{give}, \text{SEM} \\
\text{RESTR} \langle \text{GIVER} i, \text{GIVEN} j, \text{GIFT} k > \\
\text{ARG-ST} \langle X_i, Y_j, Z_k > \end{array}
\]

(40) \[
\begin{array}{c}
\text{dtv-lxm} \\
\langle \text{give}, \text{SEM} \\
\text{RESTR} \langle \text{GIVER} i, \text{GIVEN} j, \text{GIFT} k > \\
\text{ARG-ST} \langle \text{NP}_i, \text{NP}_j, \text{NP}_k > \end{array}
\]

This family of lexical sequences will give rise to another where the feature structure is of type word. These feature structures must obey the Argument Realization Principle, in consequence of which the first argument will be identified with the member of the SPR list and the remaining ARG-ST members will be identified with the two members of the verb’s COMPS list.

Note that the lexical entry in (39) includes stipulations identifying the indices of the
arguments with the role values (values of GIVER, GIVEN, and GIFT) of the lexeme's predication. In fact, much of this information is predictable on the basis of the lexeme's meaning. Though we cannot develop such an approach here, there is in fact considerable work that has proposed ways of eliminating further redundancy from lexical entries like (39). Eliminating such redundancy is one of the goals of a 'linking theory', as mentioned in Chapter 7.

8.4.3 Constant Lexemes

Let us turn now to noninflecting lexemes, that is, the various subtypes of the type \textit{const-lxm} that we have not yet considered:

(41) \begin{align*}
\text{const-lxm} \\
\text{predp-lxm} & \quad \text{argmkp-lxm} & \quad \text{adj-lxm} & \quad \text{conj-lxm} & \quad \text{det-lxm}
\end{align*}

These correspond to various kinds of lexical entries that undergo no inflection in English.\(^{14}\) Since only expressions (words or phrases) enter into grammatical structures, these lexemes must all undergo a lexical rule in order to produce (phonologically identical) words that can be of some grammatical use. We'll see this rule in Section 8.6.3.

In Chapter 7 we distinguished two kinds of prepositions - those that function as predicates and those that serve as argument markers. This distinction corresponds to the two types \textit{predicational-preposition-lexeme} (\textit{predp-lxm}) and \textit{argument-marking-preposition-lexeme} (\textit{argmkp-lxm}) in (41). Recall that in our earlier discussion we distinguished these prepositions in terms of their semantics. Only prepositions of type \textit{predp-lxm} introduce their own predication. Argument marking prepositions simply take on the INDEX and MODE value of their object. These effects are ensured by the following type constraints:

(42) a. \begin{align*}
\text{predp-lxm} : \\
\text{SYN} & \quad [\text{HEAD} \ prep] \\
\text{VAL} & \quad [\text{SPR} \ (X)] \\
\text{MOD} & \quad [\text{SPR} \ (Y)] \\
\text{SEM} & \quad [\text{MODE} \ none] \\
\text{ARG-ST} & \quad [\text{RESTR} \ (Z)] \\
\text{NP} \ NP)
\end{align*}

\(^{14}\)The type \textit{adj-lxm} arguably should be classified as a subtype of \textit{infl-lxm}, rather than as a subtype of \textit{const-lxm}, in light of the fact that many adjectival lexemes give rise to comparative and superlative forms, e.g. tall, taller, tallest. We will not pursue this matter here. Note also that the classification of lexemes into inflecting and constant is language-specific. As we saw in Problem 2 of Chapter 4, for example, determiners in Spanish inflect for agreement information.
Only predicative prepositions can be modifiers. Accordingly, \textit{argmkp-lxm} says nothing about MOD and thus inherits the default constraint [MOD / ⟨ ⟩] from \textit{lexeme}. \textit{predp-lxm}, on the other hand, overrides this constraint with [MOD ⟨ Y ⟩]. This non-empty MOD values allows these prepositions to be modifiers.\footnote{This MOD value is obviously not constrained enough, as there are things that PPs can’t modify \cite[e.g., determiners]{const-lxm} and or its instances need to say something more specific, although we won’t explore it further here.} When they appear as complements of verbs (as in (43), discussed in Chapter 7), this non-empty MOD value is irrelevant.\footnote{Note in addition that nothing in our analysis blocks the projection of subject-saturated PPs like \textit{My blanket [around me]}. As noted in Chapter 4 these occur only in restricted circumstances, e.g. as ‘absolute’ or ‘small’ clauses.}

(43) I wrapped the blanket [around me].

Note also that \textit{predp-lxm} specifies a two-place ARG-ST list and a non-empty SPR value. Once a word is built from a predicational preposition, its first argument must be identified with the SPR element, in accordance with the ARP. What plays these roles in (43) is the NP \textit{the blanket}, which is also an argument of the verb \textit{wrapped}. This is the first time we have seen one constituent serving as an argument of more than one predicate at the same time. This is a common phenomenon, however, as we will see in subsequent chapters. Developing an analysis of such cases is the topic of Chapter 12.

The argument-marking prepositions, because of the constraint in (42b), project a nonmodifying PP with an empty specifier list whose MODE and INDEX values are identified with those of the preposition’s NP object:

(44) He talks [to himself].

As described in Chapter 7, this analysis allows the objects of argument-marking prepositions to enter into binding relations with other NPs. Finally, recall that some prepositions, for example, \textit{around}, behave either as predicational or as argument-marking. Hence the following example is also well formed:

(45) I wrapped the blanket [around myself].

This pattern of optional reflexivization is now neatly accounted for by allowing \textit{around} to live a double life (via two separate lexical entries) as either a predicational or argument-marking preposition.

For the sake of completeness, we include the following three type constraints on the remaining three subtypes of \textit{const-lxm}:

\begin{center}
\begin{tabular}{|c|c|}
\hline
\textbf{SYN} & \textbf{ARG-ST} \\
\hline
\textbf{VAL} & \textbf{NP} \\
\hline
\textbf{SEM} & \textbf{SPR} \\
\hline
\textbf{ARG-ST} & \textbf{RESTR} \\
\hline
\end{tabular}
\end{center}
The constraints on the type \textit{det-lxn} are meant to accommodate the results of Chapter 6, Problem 3 - that is, that 's is a determiner that exceptionally takes an obligatory NP specifier. The types \textit{adj-lxn} and \textit{conj-lxn} will require further constraints, but we omit discussion of them here.

### 8.4.4 Lexemes vs. Parts of Speech

Some readers will have noticed that our type hierarchy posits two distinct types corresponding roughly to each of the traditional parts of speech. In addition to \textit{noun}, \textit{verb}, etc. – the subtypes of \textit{pos} introduced in Chapter 3 – we now have types like \textit{cn-lxn}, \textit{pn-lxn}, \textit{verb-lxn}, and so forth, which are subtypes of the type \textit{lexeme}. It is important to understand that these two sets of types serve rather different functions in our grammar. The subtypes of \textit{pos} specify which features are appropriate for particular categories of words and phrases. They thus serve to organize the various parts of speech our grammar has to recognize. The subtypes of \textit{lexeme}, on the other hand, introduce constraints on what combinations of feature values are possible, for example, the SHAC or the constraint that verbs require propositional mode. These typically involve argument structure (and/or valence features) as well as \textit{HEAD} features or \textit{SEM} features. Consequently, the \textit{pos} subtypes (\textit{noun}, \textit{verb}, etc.) frequently appear inside of the constraints associated with the \textit{lexeme} subtypes (\textit{noun-lxn}, \textit{verb-lxn}, etc.).

The type hierarchy simplifies our descriptions in two ways: it saves us from having to assign values to features where they would do no work, for example, \textit{PER} (person) in prepositions or \textit{CASE} in verbs; and it allows us to stipulate common combinations of feature values only once, using (default) inheritance to account for their distribution. The hierarchy contains two sets of types corresponding roughly to the traditional parts of speech then, because the hierarchy serves these two separate functions.

### 8.4.5 The Case Constraint

Up to this point, we have made no mention of \textit{CASE} specifications in our lexical type hierarchy. Thus nothing yet guarantees that NPs in English must be accusative except when they are the subject of a finite verb form. One might think this is a constraint on lexemes, but this would make certain incorrect predictions. As we will see in later chapters, certain lexical rules (such as the Passive Lexical Rule introduced in Chapter 10), have
the effect of reordering ARG-ST lists. Such reordering never results in ARG-ST-initial elements being specified as [CASE acc]. For this reason, we will treat the assignment of accusative case as a fact about words, not about lexemes. The easiest way to do this is to add the following constraint to our definition of lexical satisfaction.¹⁷

(47) Case Constraint

An outranked NP is [CASE acc].

This principle allows us to keep our constraints on verbal lexemes just as we formulated them above, with no mention of case. Thus it is unnecessary to specify lexically the accusative case for most objects, providing a significant improvement on the analysis of English case suggested in Problem 6 of Chapter 4. Notice, however, that (47) is a one-way implication: it says that certain NPs are accusative, but it says nothing about which NPs are not accusative. The nominative case, characteristic of subjects, will need to be specified in some other way (a point to which we return later in this chapter).

Finally, it must be stressed that the Case Constraint is specific to English. Many other languages exhibit far more complex case systems; see, for example, the problems on Icelandic and Wambaya in Chapter 4.

8.5 The FORM Feature

In the next section, we'll introduce the lexical rules that relate the lexemes discussed above to the inflected words they give rise to. First, however, we return to the feature FORM, which came up briefly in the discussion of imperatives in Chapter 7 (section 7.6).

8.5.1 FORM Values for Verbs

In general, different inflected words arising from the same lexeme have different distributions. In order to capture those different distributions in our grammar, we must ensure that they have different feature specifications. In many cases, this work is done by features we have already introduced. For example, singular and plural nouns differ in their NUMBER values. In the case of verbs, however, the inflected forms differ in their distributions without differing in any of the features we have posited for other uses. For example, the verb after a modal must be in the base form, the verb after auxiliary have must be a past participle, and the main verb in a sentence must be finite (past or present tense):

(48) a. Kim may
    \[
    \{ \text{leave} \\
    *\text{leaves} \\
    *\text{leaving} \\
    *\text{left} \}
    \]

b. Kim has
    \[
    \{ *\text{leave} \\
    *\text{leaves} \\
    *\text{leaving} \\
    \text{left} \}
    \]

¹⁷ Thanks to Louis Eisenberg for pointing out the possibility of this formulation of the Case Constraint.
We will use the feature FORM to distinguish between these different forms. For verbs, we will posit the following (atomic) values for the feature FORM:\(^{18}\)

(49) base The bare uninflected form, as in *Andy would eat rice*, *Andy tried to eat rice*, or *Eat rice!*

fin ‘Finite’, i.e. present or past tense, as in *Andy eats rice* or *Andy ate rice*

ptp ‘Present participle’, suffixed with *-ing*, usually following some form of *be*, as in *Andy is eating rice*

psp ‘Past participle’ (or ‘perfect participle’), the form that follows *have*, as in *Andy has eaten rice*

pass ‘Passive’, as in *Rice was eaten by Andy*

(to be discussed in Chapter 10)

Treating FORM as a head feature will allow us to get a handle on the cooccurrence restrictions illustrated in (48). As discussed in detail in Chapter 13, we treat auxiliaries like *may* or *has* as verbs that take VP complements. They specify particular form values on those VP complements, and the Head Feature Principle ensures that the FORM value of the selected VP is the same as the FORM value of the head verb inside that VP. This is illustrated in (50):

\(^{18}\)Particular researchers have made slightly different assumptions about the value for the feature FORM (or its equivalent). For example, ‘ger’ (for ‘gerund’) has sometimes been proposed for a kind of word not covered here. Like present participles, gerunds are suffixed with *-ing*, but unlike present participles, gerunds head phrases that have the distribution of NPs. The occurrences of *singing* in (i)-(iii) are present participles; those in (iv)-(vi) are gerunds.

(i) The birds are singing.
(ii) Anyone singing in class will be punished.
(iii) Ashley began singing Christmas carols in October.
(iv) Ashley’s singing Christmas carols in October annoyed Jordan.
(v) We denied singing during class.
(vi) Don’t even think about singing!

The analysis of gerunds is beyond the scope of this text. Hence, we will not consider the question of whether there should be a FORM value for gerunds.
Another benefit of treating FORM as a head feature is that it will allow us to refine our definition of the initial symbol. In Chapter 6, we gave the initial symbol as ‘S’, i.e., the combination of constraints shown in (51).

\[
\begin{array}{c}
\text{SYN} \\
\text{VAL}
\end{array}
\begin{array}{c}
\text{HEAD} \\
\text{verb}
\end{array}
\begin{array}{c}
\text{COMPS} \\
\{ \}
\end{array}
\begin{array}{c}
\text{SPR} \\
\{ \}
\end{array}
\]

We would now like to add the constraint that only finite Ss can be stand-alone sentences. We can achieve this by adding the specification \([\text{FORM fin}]\) to our definition of the ‘initial symbol’, which specifies which sentences can serve as independent utterances:

\[
\begin{array}{c}
\text{SYN} \\
\text{VAL}
\end{array}
\begin{array}{c}
\text{HEAD} \\
\text{verb}
\end{array}
\begin{array}{c}
\text{COMPS} \\
\{ \}
\end{array}
\begin{array}{c}
\text{SPR} \\
\{ \}
\end{array}
\begin{array}{c}
\text{FORM fin}
\end{array}
\]

Since FORM is a head feature, the only Ss that are \([\text{FORM fin}]\) are those which are ultimately headed by verbs that are \([\text{FORM fin}]\), as illustrated in (53):

\[\text{The one exception is imperatives, which are finite Ss not headed by a finite verb. This comes about because the Imperative Rule is a non-headed rule and it changes the FORM value. In this sense, imperative sentences are not in fact headed by anything.}\]
The previous section argued that FORM is best treated as a HEAD feature. The current version of our Coordination Rule (last discussed in Chapter 5) does not identify the HEAD values of the conjuncts with each other or with the mother. It turns out that this makes incorrect predictions. Where verbs select for VPs of a particular form, that selection holds even if the complement is a coordinated VP:

\[(53)\]

\[
S \\
| \quad \text{HEAD} \ \square \\
| \\
NP \quad \text{VP} \\
| \\
Kim \quad \text{uses} \\
| \\
V \quad \text{NP} \\
| \\
\text{HEAD} \ \square \text{FORM} \ \text{fin} \\
| \\
likes \quad \text{Sandy}
\]

### 8.5.2 FORM and Coordination

The previous section argued that FORM is best treated as a HEAD feature. The current version of our Coordination Rule (last discussed in Chapter 5) does not identify the HEAD values of the conjuncts with each other or with the mother. It turns out that this makes incorrect predictions. Where verbs select for VPs of a particular form, that selection holds even if the complement is a coordinated VP:

\[(54)\]

<table>
<thead>
<tr>
<th>Dana helped Leslie</th>
</tr>
</thead>
<tbody>
<tr>
<td>move</td>
</tr>
<tr>
<td>*moves</td>
</tr>
<tr>
<td>*moving</td>
</tr>
<tr>
<td>*moved</td>
</tr>
<tr>
<td>pack and move</td>
</tr>
<tr>
<td>*packs and move</td>
</tr>
<tr>
<td>*pack and moves</td>
</tr>
<tr>
<td>*packs and moves</td>
</tr>
<tr>
<td>*packing and move</td>
</tr>
<tr>
<td>*pack and moving</td>
</tr>
<tr>
<td>*packing and moving</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Likewise, stand-alone coordinate sentences must contain a finite verb as the head of each conjunct:

\[(55)\]

a. Dana walked and Leslie ran.

b.*Dana walking and Leslie ran.

c.*Dana walked and Leslie running.

d.*Dana walking and Leslie running.
In order to capture these facts, we add a constraint to our Coordination Rule that identifies the FORM values of the conjuncts with that of the mother.\(^{20}\) In making this revision, the Coordination Rule has almost reached its final form. We will revisit it once more in Chapter 14.

(56) Coordination Rule (Chapter 8 Version)

\[
\begin{array}{c}
\text{VAL} & \square \\
\text{HEAD} & \left[\text{FORM } \square\right] \\
\text{IND} & s_0
\end{array} \rightarrow \\
\begin{array}{c}
\text{VAL} & \square \\
\text{HEAD} & \left[\text{FORM } \square\right] \ldots \\
\text{IND} & s_1 \quad \text{IND} & s_n
\end{array} \quad \begin{array}{c}
\text{HEAD} & \text{conj} \\
\text{IND} & s_0 \\
\text{RESTR} & \left\{\text{ARGS} \left\{s_1 \ldots s_n\right\}\right\}
\end{array} \quad \begin{array}{c}
\text{VAL} & \square \\
\text{HEAD} & \left[\text{FORM } \square\right] \\
\text{IND} & s_n
\end{array}
\]

Adding FORM identity constraints to the Coordination Rule raises two important (and related) points. The first is that FORM must now be appropriate for all pos types that can be coordinated. If it weren’t, then expressions with pos types that don’t bear the FORM feature could never be compatible with the rule. The second point to note is that the FORM values we have posited so far (prp, psp, pass, etc.) are only appropriate for verbs.\(^{21}\) This means that the Coordination Rule no longer incorrectly allows the coordination of, say, NP and S (cf. section 4.7 of Chapter 4):

(57) Dana walked and Kim.

Since FORM must be appropriate for all parts of speech that can coordinate, we can use the FORM identity condition to impose the requirement that conjunct daughters must have the same part of speech, but we can do so without identifying their HEAD values. (Recall from section 4.7 of Chapter 4 that requiring HEAD identity is too strong, because it disallows conjuncts with different AGR values.) We do this by positing distinct FORM values for each part of speech. Nouns will be [FORM nform], adjectives will be [FORM aform], and so forth. For many lexical classes, we can guarantee these correlations between part-of-speech types and FORM values in a general way by stating defeasible constraints on the relevant subtype of pos. (58) is such a constraint:

(58) \(\text{noun} : \left[\text{FORM } / \text{nform}\right]\)

This constraint is defeasible, as we will use special FORM values for certain nouns and pronouns in the treatment of expletives and idiomatic expressions that we present in Chapter 11. We will also posit special values of FORM to distinguish among prepositions in our account of selectional dependencies between verbs and prepositions (see Chapter 10). But there is no need to assume a FORM value ‘vform’ or to give a default FORM value to verbs, as all inflected forms of verbs are given a specific FORM value by one of the inflectional rules discussed in the next section.

\(^{20}\)Note that we have abbreviated the feature paths in this statement of the rule.

\(^{21}\)This is true of the grammar so far. However, in Chapter 11 we will see that complementizers can take some of the verbal FORM values. We’ll return to what this means for the analysis of coordination in Problem 8 of Chapter 11.
8.6 Inflectional Rules

The lexical rule is a mechanism for further reducing redundancy and stipulation in the lexicon by stating systematic regularities that hold between lexical entities, e.g. between lexemes and the words that are ‘realizations’ of those lexemes. Lexical rules are used to explicate the predictable relations among lexical entities, such as inflected forms of verbs and nouns.

It is traditional to think of words (or at least certain kinds of words) as being built up from smaller units through the addition of affixes. We have followed this tradition by using our notion of types to distinguish lexeme from word. For most nouns and verbs, we will assume that there is only one basic lexical entry. As explained in the previous section, each such lexical entry describes a family of lexical sequences. We then characterize all the nominal and verbal words in terms of lexical rules that relate the basic lexical sequences to others whose second member is a feature structure of type word.

Although it is intuitive, as well as traditional, to think of a lexical rule as a process that takes lexemes (or words) as input and gives distinct lexical entities as output, it is not necessary to introduce a new kind of device to capture the essential insights of lexical rules. In fact, lexical rules can be modeled as feature structures of a special type, which we’ll call lexical-rule (l-rule). Feature structures of this type specify values for the features INPUT and OUTPUT. There are a number of advantages to be derived from modeling lexical rules in this way. For example, they can be organized into a type hierarchy, with common properties factored into constraints on common supertypes. This is particularly attractive, as languages that have more complicated morphological paradigms require families of lexical rules that have many properties in common. This is true, for example, of the lexical rules that are required for the Spanish verb paradigms we considered at the beginning of this chapter.

A second advantage of modeling lexical rules as feature structures is that we can use defeasible identity constraints on the values of the features INPUT and OUTPUT. A defeasible identity constraint can guarantee that constraints holding of a lexical rule input are carried over to the rule’s output, by default. This will let us streamline the formulation of lexical rules, allowing our grammar to stipulate only those properties that add or alter specific pieces of information.

We can thus think of a lexical rule as a feature structure that corresponds to a particular lexical relation holding between pairs of lexical sequences. We will here consider two types of l-rule: inflectional-rule (i-rule) and derivational-rule (d-rule) organized into the following type hierarchy:

\[ l\text{-rule} \rightarrow i\text{-rule} \rightarrow d\text{-rule} \]

---

\[ ^{22}\text{There have been many proposals for how to formulate lexical rules, ranging from ‘metadescription’ approaches that apply generatively to map lexical entries into lexical entries and ‘redundancy rule’ approaches that treat them as stating generalizations that hold over a pre-existing set of entries.} \]

\[ ^{23}\text{Another subtype of i-rule will be introduced in Chapter 11.} \]
All feature structures of type *l-rule* obey the following constraint:

\[\begin{align*}
\text{INPUT} & \quad \text{l-sequence} \left[ X, \left[ \text{SEM} / I \right] \right] \\
\text{OUTPUT} & \quad \text{l-sequence} \left[ Y, \left[ \text{SEM} / I \right] \right]
\end{align*}\]

What (60) says is that both the input and output of a lexical rule are lexical sequences (see page 242) and that the SEM values of the lexical rule's input and output are identical, again by default.

But inflectional rules obey stronger constraints, namely, those we formulate as in (61):

\[\begin{align*}
\text{INPUT} & \quad \left[ X, \left[ \text{lexeme} \right] \right] \\
\text{OUTPUT} & \quad \left[ Y, \left[ \text{word} \right] \right]
\end{align*}\]

(61) says that the input of an inflectional rule must be of type *lexeme* and that its output must be of type *word*. (61) also requires that the input and output share both SYN and ARG-ST values. Note that this last requirement allows inflectional rules to add constraints to the output, as long as they are consistent with constraints placed on the input lexeme. However, (61) guarantees that inflectional rules perform no ‘destructive’ changes to the SYN or ARG-ST value of a lexeme, for this would contradict the indicated identity constraints. We will illustrate this property of inflectional rules in the following section. We take up derivational rules in Section 8.7 and in subsequent chapters.

We will use lexical rules to specify the set of word structures allowed by our grammar. This can be done rather simply by requiring the mother nodes in all word structures to be labelled by feature structures (of type *word*) that are the second member of the OUTPUT of some lexical rule. Lexical rule descriptions such as those we will posit in the following sections are each satisfied by a family of feature structures. But since such feature structures are resolved, so are their values for the features INPUT and OUTPUT. Our theory of lexemes provides an account of which feature structures of type *lexeme* are well-formed, and hence constraints like the ones shown in (63) below relate any lexical sequence containing such a lexeme to a lexical sequence containing an appropriately resolved feature structure of type *word*. Once we formulate the constraints on our particular lexical rules correctly, their outputs will be precisely the set of lexical sequences we need to license the word structures we want. Finally, the mother of a word
structure must be fully resolved because the only satisfiers of a lexical rule are lexical sequences containing feature structures, which are fully resolved by definition. Hence the value for the feature OUTPUT in a lexical rule is also fully resolved.

8.6.1 Rules for Common Noun Inflection

Once we have the type constraints just outlined, we may introduce specific instances—more precisely, feature structures that instantiate the type i-rule. Let's consider first the inflectional rule that relates common noun lexemes to their singular word realizations, i.e. the rule that is responsible for words like dog and water. These words are specified as (third-person) singular, but otherwise they contain just the (phonological, syntactic and semantic) information that they inherit from the lexeme they are related to. Given this, we can formulate the rule we need as shown in (62), where the form of the output word is required to be identical to that of the input lexeme.\(^{24}\)

\(\text{(62) \ Singular Noun Lexical Rule} \)

\[
\begin{align*}
\text{i-rule} \\
\text{INPUT} & \langle 1, \text{cn-lxm} \rangle \\
\text{OUTPUT} & \langle 1, \text{word} [\text{SYN} [\text{HEAD} [\text{AGR} [\text{NUM sg}]]]] \rangle
\end{align*}
\]

Given that (62) describes a feature structure that is an instance of the type i-rule (constrained as shown in (60) and (61)), it follows from the theory of constraint inheritance sketched above that the lexical rule is constrained as follows:

\(\text{(63) \ Singular Noun Lexical Rule (with inherited constraints)} \)

\[
\begin{align*}
\text{i-rule} \\
\text{INPUT} & \langle 1, \text{word} \rangle \\
\text{OUTPUT} & \langle 1, \text{word} [\text{SYN} [\text{HEAD} [\text{AGR} [\text{NUM sg}]]]] \rangle
\end{align*}
\]

Notice that nothing in (62) contradicts any of the defeasible identity constraints in (60) and (61). Hence all those constraints remain in effect in (63). The set of constraints shown in (63) is exactly what we get as the result of combining the defeasible constraints

\(^{24}\text{It is thus an ‘accident’ of English morphology that singular nouns, unlike plural nouns, have no inflectional ending.} \)
in (60) with the inviolable constraints in (61) and (62).\(^{25}\)

Let us consider a simple example. In (33) above, we explained how our grammar gives rise to the family of lexical sequences represented by the following:

\[
\text{(64)} \quad \begin{array}{c}
\text{cntn-lzn} \\
\text{SYN} \\
\text{VAL} \\
\text{SEM} \\
\text{HEAD} \\
\text{AGR} \\
\text{PER 3rd} \\
\text{SPR} \\
\text{COUNT} \\
\text{AGR} \\
\text{COMPS} \\
\text{MODE} \\
\text{ref} \\
\text{INDEX} \\
\text{i} \\
\text{RESTR} \\
\text{RELN} \\
\text{dog} \\
\text{INST} \\
\text{i}
\end{array}
\]

And any of the lexical sequences in (64) is a possible value of the feature INPUT in a feature structure that satisfies the Singular Noun Lexical Rule (with its inherited constraints - shown in (63) above). If the INPUT of (63) is resolved to such a lexical sequence, the following will be the family satisfying the value of the feature OUTPUT:

\[
\text{(65)} \quad \begin{array}{c}
\text{word} \\
\text{SYN} \\
\text{VAL} \\
\text{SEM} \\
\text{HEAD} \\
\text{AGR} \\
\text{PER 3rd} \\
\text{SPR} \\
\text{COUNT} \\
\text{AGR} \\
\text{COMPS} \\
\text{MODE} \\
\text{ref} \\
\text{INDEX} \\
\text{i} \\
\text{RESTR} \\
\text{RELN} \\
\text{dog} \\
\text{INST} \\
\text{i}
\end{array}
\]

\(^{25}\text{Note, however, that if an input were specified as [NUM pl] (plausible examples might be scissors or pants), then it would fail to undergo this lexical rule. That is, there could be no relation between the input lexical sequence and any output lexical sequence that satisfied the constraint specified in (63).}\)
The feature structures in (65) are licensed as lexical sequences whose second member is a feature structure of type \textit{word}.\footnote{In what follows, we will loosely talk of lexical rules relating lexemes to words, etc.} By the informal definition given in the introduction to this section, these \textit{words} can be used as the daughters of phrase structure rules to build phrases and sentences. In Chapter 9, we will give a reformulation of lexical licensing that refers to OUTPUT values of lexical rules.

In the remainder of this section, we will briefly introduce some of the particular lexical rules we posit to relate lexemes to words. In the next section, we discuss briefly \textsc{derivational rules}, which relate lexemes to lexemes. In Chapter 11, we will also countenance lexical rules that relate words to words.

The next lexical rule to consider is the rule that maps nominal lexemes into lexical entries for their corresponding plural forms:

(66) Plural Noun Lexical Rule

\[
\begin{array}{l}
\text{INPUT} \quad \langle \text{cntn-lexm} \rangle \\
\text{OUTPUT} \quad \langle \text{NPL} \left( \langle \text{cntn-lexm} \rangle \right) \rangle
\end{array}
\]

Here, \text{NPL} is a morphological function that applies to a nominal base in English, giving its plural form. This function is sketched in (67):

(67)

\[
\begin{array}{|l|l|}
\hline
\text{X} & \text{NPL}(X) \\
\text{child} & \text{children} \\
\text{ox} & \text{oxen} \\
\text{woman} & \text{women} \\
\text{fish} & \text{fish} \\
\text{index} & \text{indices} \\
\text{...} & \\
\text{(otherwise)} & \text{X-s} \\
\hline
\end{array}
\]

There are various issues that arise in connection with such inflectional functions, e.g. how best to accommodate subregularities and similarities across different morphological functions, but we will steer clear of these issues here.

The lexical rule sketched in (66) inherits constraints from the types \textit{i-rule} and \textit{l-rule}. The combination of (66) and (60) and (61) is indicated in (68):
The Plural Noun Lexical Rule thus guarantees that for every count noun lexeme\(^{27}\) there is a corresponding plural noun word with identical SYN, SEM, and ARG-ST values, whose form is determined by the function \(F_{NPL}\). The requirement that the input be \(cntn/-lxm\) keeps the rule from applying to mass nouns like furniture, so that there is no word */furniture*. The Plural Noun Lexical Rule thus allows for lexical sequences like (69):\(^{28}\)

\[\text{(69)}\]

\[
\text{INPUT} \begin{cases}
\text{SYN} & \alpha \\
\text{SEM} & \alpha \\
\text{ARG-ST} & \beta
\end{cases}
\]

\[
\text{OUTPUT} \begin{cases}
\text{word} & \text{F}_{NPL} \begin{cases}
\text{SYN} & \begin{cases}
\text{HEAD} & \beta \\
\text{AGR} & \begin{cases}
\text{NUM} & \text{pl}
\end{cases}
\end{cases}
\end{cases}
\end{cases}
\]

\[\text{SEM} & \alpha \\
\text{ARG-ST} & \beta
\end{cases}
\]

Other than those that might be lexically restricted to be singular.

\(^{27}\) A complete formulation of both lexical rules discussed so far would require the introduction of a fundamental difference between the semantics of singular and plural nouns. Unfortunately, a semantic analysis of singular and plural nouns – which would have to include a treatment of the count/mass distinction – is beyond the scope of this book.
8.6.2 Rules for Inflected Verbal Words

We posit additional lexical rules for the various inflected forms of verbs, beginning with the rule for the 3rd-singular present form:

(70) 3rd-Singular Verb Lexical Rule

\[
\text{i-rule}
\]

\[
\text{INPUT} \quad \left\langle [\text{verb-lxm}], \begin{cases} \text{SEM} \left[\text{RESTR A}\right] \end{cases} \right\rangle
\]

\[
\text{OUTPUT} \quad \left\langle \text{F}_{3\text{SG}}[\text{A}], \begin{cases} \text{SYN} \left[\text{HEAD } \left[\text{FORM fin}\right]\right], \text{SEM} \left[\text{RESTR A} \oplus \ldots\right], \text{ARG-ST} \left[\begin{cases} \text{CASE } \text{nom}, \ldots \end{cases} \right] \right\rangle
\]

As with the Plural Noun Lexical Rule, we have glossed over the morphological component of the 3rd-Singular Verb Lexical Rule by simply giving it a name, \(F_{3\text{SG}}\).

The semantic effect of this rule is to preserve the basic semantics of the input, but add the tense information. That is, MODE and INDEX are unchanged, but a predication representing tense is added to the RESTRICTION. Predications of this type will be suppressed here and throughout, with \(\ldots\) standing in.\(^{29}\) What the rule in (70) says, then, is that for any verbal lexeme, there is a corresponding finite verb (a word) that takes a third-person singular subject whose CASE value is nominative. Further, the morphology and semantics of the latter verb are systematically related to those of the input lexeme.

We turn next to the rule that licenses finite verbs with subjects other than third-person singular NPs. Because the type distinction we have drawn between the AGR values 3sing and non-3sing already distinguishes third-singular NPs from all others, this rule is almost identical to the last one, as shown in (71):

\(\text{\footnotesize{\text{footnote}}}
\)

\(\text{\footnotesize{\text{footnote}}}
\)
The only differences between (71) and (70) are: (i) no change in morphology is introduced, and (ii) the nominative subject of the output must have a non-3sing AGR value (see Chapter 4, section 4.6 for further discussion). Outputs of this rule, for example the one shown in (72), sanction word structures that can never combine with a third-person singular subject:

As with the 3rd-Singular Verb Lexical Rule, the semantics of the output is systematically related to the semantics of the input.

The two rules just discussed license the present tense forms of verbs. The next rule
creates lexical sequences for the past tense forms. English makes no distinction between singular and plural in past tense forms (aside from was vs. were); hence only one rule is needed:

(73) Past-Tense Verb Lexical Rule

\[
\begin{align*}
\text{i-rule} \\
\text{INPUT} & \langle [\text{SEM}], [\text{RESTR}] \rangle \\
\text{OUTPUT} & \langle F_{\text{PAST}}(\text{SEM}), [\text{SYN}], [\text{ARG-ST}] \rangle \\
& \quad \langle [\text{HEAD}], [\text{FORM}], \rangle \\
& \quad \langle [\text{SEM}], [\text{RESTR}], \rangle \\
& \quad \langle [\text{ARG-ST}], [\text{CASE}], \rangle \\
& \quad \langle [\text{FORM}], \rangle \\
& \quad \langle [\text{HEAD}], \rangle \\
& \quad \langle [\text{SEM}], \rangle \\
& \quad \langle [\text{RESTR}], \rangle \\
& \quad \langle [\text{ARG-ST}], \rangle \\
& \quad \langle [\text{CASE}], \rangle \\
& \quad \langle [\text{FORM}], \rangle \\
& \quad \langle [\text{HEAD}], \rangle \\
& \quad \langle [\text{SEM}], \rangle \\
& \quad \langle [\text{RESTR}], \rangle \\
& \quad \langle [\text{ARG-ST}], \rangle \\
& \quad \langle [\text{CASE}], \rangle \\
& \quad \langle [\text{FORM}], \rangle \\
& \quad \langle [\text{HEAD}], \rangle \\
& \quad \langle [\text{SEM}], \rangle \\
& \quad \langle [\text{RESTR}], \rangle \\
& \quad \langle [\text{ARG-ST}], \rangle \\
& \quad \langle [\text{CASE}], \rangle \\
\end{align*}
\]

(73) makes use of a function \(F_{\text{PAST}}\) to account for the morphological relation between verbal lexemes and their past tense forms; in most cases, this consists of suffixing \(-ed\), though there are many exceptions (such as \(sleep/slept\), \(eat/ate\), and \(put/put\)).

Like the lexical rules for present tense verbs, (73) requires its subject to be nominative (to rule out examples like \(\ast \text{Me slept}\)); but unlike the present tense rules, it puts no number or person restrictions on the subject, since English past tense verbs exhibit no agreement with their subjects. The semantic effect of the rule is parallel to that of the two present tense rules, though the required semantics is different.\(^{31}\)

8.6.3 Uninflected Words

Finally, we need a trivial lexical rule for noninflecting lexemes:

\(\ast\)Of course, something must be said about this exception and about the first-person singular form \(am\). The fact that \(be\) makes finer distinctions among its verb forms than other verbs does not justify making these distinctions throughout the rest of the verbal system in English. Rather, it is more parsimonious to make \(be\) an exception to some of these lexical rules, and to stipulate the individual forms in the lexicon or to post highly specialized lexical rules for the forms of \(be\). (The latter course may be desirable because, as we shall see at several points in the rest of this book, there appear to be several different \(be\) lexemes in English.) We will not go into the question of what kind of formal machinery to use to specify that certain lexical entries are exceptions to certain lexical rules, though some such mechanism is surely needed irrespective of \(be\).

The inflectional paradigm of \(be\) looks quite confusing at first, with one form \(am\) that goes only with first-person subjects and others \(are, were\) that go only with subjects that are second or plural. The situation looks a bit less arbitrary if we make use of the hierarchy of subtypes of \(non-\text{sing}\) introduced in Chapter 4. That hierarchy makes available a type \(1\text{sing}\) that is the AGR value we need for the specifier of \(am\). It also provides a type \(non-1\text{sing}\) encompassing just second-person and plural AGR values (that is, it excludes just the first-person singular and third-person singular values). This is precisely the AGR value we need for the specifier of \(are\) and \(were\). To accommodate \(was\), we would need either two lexical entries (one requiring a \(1\text{sing}\) specifier and another requiring a \(3\text{sing}\) specifier) or a lexical entry with a disjunctive SPR value.

\(\ast\)In the same spirit as the representation of present tense sketched in note 29, we could represent past tense by adding a ‘temporal precedence’ relation to the RESTR value. That is, the situation denoted by the index of the verb temporally precedes the time of utterance if the verb is in the past tense. Again, this is only an approximation of the semantics of the English past tense.
This rule does nothing except allow the requisite words to be licensed from homophonous lexemes. The SYN, SEM and ARG-ST values of these words will be identical to those of the corresponding lexeme. This already follows from the inheritance of the identity constraints in (60) and (61).

8.7 Derivational Rules

Each of the lexical rules in the previous section maps lexical sequences of type lexeme into sequences of type word. We have followed tradition in calling these inflectional rules. It is also traditional to distinguish these from another kind of lexical rule (called a derivational rule) that relates lexemes to lexemes (or, in our system, lexical sequences of the appropriate kind to other such lexical sequences). Derivational rules (d-rules) are appropriate when the addition of a prefix or suffix creates a new lexical item that can itself undergo inflectional rules. We will assume that d-rules are constrained as follows:

\[
\begin{align*}
\text{d-rule:} & \quad \text{INPUT} \quad X : \left[ \text{lexeme SYN /} \right] \\
& \quad \text{OUTPUT} \quad Y : \left[ \text{lexeme SYN /} \right]
\end{align*}
\]

Let us consider as a agentive nominalizations as a first example. Noun lexemes like driver or eater might be derived by the following lexical rule:

\[
\begin{align*}
\text{d-rule} & \quad \text{INPUT} \quad \left[ \text{verb-lxm SEM INDEX s} \right] \\
& \quad \text{OUTPUT} \quad \left[ \text{cntn-lxm SEM INDEX i} \right]
\end{align*}
\]

Here the function \( F_{er} \) adds the appropriate suffix to the form of the rule output. The input involves a verbal lexeme whose subject’s index \( i \) is identified with the index of the
nominal output. Note that the change in type from verb-lxm to cn-lxm has many side 
effects in terms of values of head features and in terms of the MODE value within the 
semantics. However, the RESTR value remains unchanged, as the information present in 
the input is compatible with the type constraints associated with the output type.

The ARG-ST values in (76) deserve some comment. The input must be a strictly 
transitive verb. Thus we correctly rule out agent nominals of such verbs as rely or put:

(77)  a. *the relier (on Sandy)
     b. *the putter (of books) (on the table)

The output, like other common nouns, takes a determiner. In addition, the output’s 
SPR value (and hence the first member of the ARG-ST list (Y)) will be a [COUNT +] 
determiner, according to constraints on the type cn-lxm. And the agent nominal may 
take a PP complement whose object is identified with the object of the input verb. This 
is for agent nominals such as the discoverer of oxygen and a builder of bridges.33

Consider, for example, the lexical entry for the verbal lexeme drive, the semantics of 
which is a proposition whose RESTR value contains a driving predication, with the role of 
driver assigned to the referent of the verb’s subject. Applying the Agent Nominalization 
Lexical Rule to this lexeme yields an entry for the nominal lexeme driver, whose index is 
restricted to be the driver in a driving predication (since the RESTR value is unchanged):

(78)

This lexeme can now undergo both our nominal lexical rules, and so we derive l-sequences 
for both the singular noun word driver and its plural analog drivers.

There are further semantic constraints that must be placed on our derivational rule, 
however. For example, the subject in the input verb has to be sufficiently agentive —

33We provide no account here of intransitive agentive nouns like jumper, runner, diver, etc.
33Notice that in formulating this rule, we have used the FORM value ‘of’ to indicate that the prepo-
  sition heading this PP must be of. We return to the matter of FORM values for prepositions in Chapter 10.
that is, it must play an active (usually volitional) role in the situation. That’s why nominalizations like knower or resemblor sound funny. But the formulation in (78) is a reasonable first pass at the problem, and it gives you an idea of how phenomena like this can be analyzed within our framework.

There are many other cross-categorial relations that work this way in English. Noun lexemes, both common and proper, can be converted into verbal lexemes:

(79)  a. Sandy porched the newspaper without difficulty.
    b. The senator houdinièd his way out of the accusations.
    c. They have been computerizing me to death all morning.

This kind of derivation without morphological change, an instance of what is often called zero derivation, could be handled by one or more derivational rules.

Derivational rules are also a traditional way of approaching the problem of valence alternations, that is, the fact that many verbs allow systematically related valence patterns. Among the most famous of these is the dative alternation illustrated in (80) – (81).

    b. Jan gave a book to Dale.

    b. Jan handed a book to Dale.

Rather than list entries for two distinct verbal lexemes for give, hand, and a family of related elements, it makes much more sense to list only one (with one of the two valence patterns fixed) and to derive the other by a derivational rule. Note however, that there are certain other verbs or particular idiomatic uses that appear in only one of the two valence patterns:

(82)  a. Kris donated a book to the library.
    b.*Kris donated the library a book.

(83)  a. Dale gave Brooke a hard time.
    b.? Dale gave a hard time to Brooke.

These underline once again the need for a theory of exceptions to lexical rules and lexical irregularity.

Other famous examples of valence alternation are illustrated in (84)–(88).

(84)  a. The police sprayed the protesters with water.
    b. The police sprayed water on the protesters. (‘spray/load’ alternations)

(85)  a. The students drove cars.
    b. These cars drive easily. (‘middle’ uses)

(86)  a. Pat sneezed.
    b. Pat sneezed the napkin off the table. (‘caused motion’ uses)

(87)  a. The horse kicked me.
    b. The horse kicked me black and blue. (‘resultative’ uses)
(88) a. They yelled.
    b. They yelled their way into the meeting. (the ‘X’s way’ construction)

All these patterns of valence alternation are governed by both semantic and syntactic constraints of the kind that could be described by finely tuned lexical rules.

Finally, we will use derivational rules to treat verbal participles like those illustrated in (89) (and discussed earlier):

(89) a. Kim is *standing* here.
    b. Sandy has *eaten* dinner.

The *d*-rules we need are formulated as follows:

(90) Present Participle Lexical Rule

\[
\text{d-rule}
\]

\[
\text{INPUT } \langle \text{verb-lxms} \rangle, \left[ \begin{array}{l}
\text{SEM} \ [\text{RESTRI} \ \text{X}]
\end{array} \right] \\
\left[ \begin{array}{l}
\text{part-lxms}
\end{array} \right]
\]

\[
\text{OUTPUT } \langle \text{F}_{\text{ing}}(\text{X}) \rangle, \left[ \begin{array}{l}
\text{SYN} \ [\text{HEAD} \ [\text{FORM} \ \text{ppp}]]
\end{array} \right] \\
\left[ \begin{array}{l}
\text{SEM} \ [\text{RESTRI} \ \text{X} \oplus \ldots]
\end{array} \right]
\]

(91) Past Participle Lexical Rule

\[
\text{d-rule}
\]

\[
\text{INPUT } \langle \text{verb-lxms} \rangle, \left[ \begin{array}{l}
\text{SEM} \ [\text{RESTRI} \ \text{X}]
\end{array} \right] \\
\left[ \begin{array}{l}
\text{part-lxms}
\end{array} \right]
\]

\[
\text{OUTPUT } \langle \text{F}_{\text{ppp}}(\text{X}) \rangle, \left[ \begin{array}{l}
\text{SYN} \ [\text{HEAD} \ [\text{FORM} \ \text{ppp}]]
\end{array} \right] \\
\left[ \begin{array}{l}
\text{SEM} \ [\text{RESTRI} \ \text{X} \oplus \ldots]
\end{array} \right]
\]

Note that the outputs of these rules belong to the type *participle-lexeme* (*part-lxms*), which is a subtype of *const-lxms* in our grammar. Thus participles undergo no further inflectional processes. This is, in essence, an arbitrary fact of English, as participles do undergo inflection in other Indo-European languages, for example in French:

(92) a. *Il y est allé*
    *he there has gone-m.sg
    ‘He went there.’

b. *Ils y sont allés*
    *they there have gone-m.pl
    ‘They(masc.) went there.’
c. *Elle y est allée*
   she there has gone-f.sg
   ‘She went there.’

d. *Elles y sont allées*
   they there have gone-f.pl
   ‘They(fem.) went there.’

Such examples show that the lexical rule for past participles in French must be derivational (that is, lexeme-to-lexeme); otherwise, participles could not serve as inputs to the inflectional rules responsible for the agreement suffixes. Our formulation of the English participle rules as derivational minimizes the differences between the grammars of English and French in this regard.\(^{34}\)

In Chapter 10, we will extend our account of participle-lexemes to include passive participles as well.

### 8.8 Conclusion

An important insight, going back at least to Saussure, is that all languages involve arbitrary (that is, unpredictable) information. Most clearly, the association between the forms (sounds) and meanings of words is purely conventional, in the vast majority of cases. A grammar of a language must list these associations somewhere. The original conception of the lexicon in modern linguistics was simply as the repository of such arbitrary information.

This conception did not last long, however. Beginning in the early years of transformational grammar, linguists began enriching their conception of the lexicon to include information that was not idiosyncratic to individual words. This trend continued in a great deal of research carried out within a variety of grammatical frameworks.

In this text, we have to some extent recapitulated this history. We began with context-free grammar in which the lexicon contained only idiosyncratic information, and we gradually enriched our lexical representations, including more and more information—much of it systematic and predictable—about the grammatical and semantic properties of words. Indeed, most of the information needed to determine the well-formedness of sentences is now encoded into our lexical entries.

With the increased expressiveness and concomitant complexity of lexical entries came a need to express succinctly certain generalizations about words. In this chapter, we have examined two formal mechanisms for capturing such generalizations. Structuring the lexicon as a hierarchy of types through which constraints are inherited (an innovation of the mid-1980s) has made it possible to factor out information common to many lexical entries, thereby greatly reducing lexical redundancy. By allowing certain type constraints to be defeasible, we have encoded default values for features, while still allowing for lexical idiosyncrasy. The second mechanism, the lexical rule, is an older idea, going back to work in transformational grammar of the 1970s. We will make considerable use of lexical rules in subsequent chapters. In fact, many of the phenomena that provided the motivation for transformations in the 1950s and 1960s can be reanalyzed in our theory using lexical rules.

\(^{34}\)We know of no evidence strictly from English for choosing between a derivational formulation and an inflectional formulation of the past and present participle rules.
rules. These include the passive construction – the topic of Chapter 10 – and many of the properties of the English auxiliary verb system, which we treat in Chapter 13.

8.9 Further Reading

8.10 Problems

Problem 1: Parts of Speech and Types
What would happen if we tried to eliminate the pos subtypes like noun and verb? To answer this, you will need to consider where the features currently associated with the pos subtypes would have to be declared, and what consequences this would have for our feature structures. Be explicit.

Problem 2: Plural and Mass NPs Without Specifiers
There is a problem with our treatment of common nouns. The type cn-lxm requires common nouns to have nonempty SPR lists, and this requirement is preserved in the Plural Noun Lexical Rule. Similarly, the type massn-lxm inherits the constraint on the SPR, and this constraint is preserved when these nouns undergo the inflectional rules. This treatment makes the wrong predictions: specifiers are optional for plural nouns and mass nouns.

A. Give examples showing, for one plural noun and one mass noun, that the specifier is optional (i.e. permitted but not obligatory).

Two obvious approaches to this problem are the following:

(i) allow empty SPR lists in the lexical entries for plural and mass nouns; or
(ii) introduce a new grammar rule to account for NPs with plural or mass heads and no specifiers.

Alternative (i) would involve modifying the Plural Noun Lexical Rule, as well as the type massn-lxm to make the first member of the ARG-ST list optional.\(^{35}\)

The rule in alternative (ii) is analogous to the Imperative Rule given in Chapter 7, in that it would have only one constituent on the right hand side, and its function would be to license a constituent without a specifier, although its daughter has a nonempty SPR list.

It turns out that alternative (i) makes incorrect predictions about prenominal modifiers (see Problem 1 of Chapter 5). We want adjectives like cute to modify plural nouns even when they don’t have specifiers:

(iii) Cute puppies make people happy.

Under alternative (i), in order to generate (iii), we would have to allow adjectives like cute to modify NPs (i.e. expressions that are \([\text{SPR} \{ } \)\]). If we do that, however, we have no way to block (iv):\(^{36}\)

(iv) *Cute the puppies make people happy.

Alternative (ii), on the other hand, would allow cute to always modify a NOM (\([\text{SPR} \{ D \} ]\)) constituent. A NOM, modified or otherwise, could either be the daughter of the non-branching rule, or the head daughter of the head-specifier rule.

\(^{35}\)This would require making the constraint on the ARG-ST of cn-lxm defeasible.

\(^{36}\)There are also technical problems with making alternative (i) work with the ARP.
B. Formulate the rule required for alternative (ii). Be as precise as you can.

[Hint: The trickiest part is formulating the rule so that it applies to both plural count nouns and mass nouns, while not applying to singular count nouns. You will need to include a disjunction in the rule. The SPR list of the head daughter is a good place to state it, since the three types of nouns differ in the requirements they place on their specifiers.]

Problem 3: -s
In most cases, $F_{SSG}$ has the same effect as $F_{NPL}$, namely, that of suffixing -s. In fact, both suffixes have multiple pronunciations, and the conditions under which they are pronounced like $s$, like $z$, or like $iz$ are identical. (They depend on phonological properties of the preceding sound.) Nevertheless, these two morphological functions are not identical. Why?

[Hints: 1. Remember that a function is single-valued, i.e. it specifies only one output for each input. 2. Consider elements that can be used as both nouns and verbs.]

⚠️ Problem 4: Another Lexical Rule

A. Write the lexical rule that is needed to generate the base form of verbs in English. You should be able to specify the syntactic effects of this rule quite precisely and straightforwardly. For the purposes of this problem, you may ignore any semantic effect the rule might have.

B. The base form lexical rule doesn’t change the phonological form. Does this mean that we don’t actually need this rule? Why or why not?

C. Extra Credit: List a few verbs that are exceptional in not undergoing this lexical rule. [Hint: Remember that $FORM_{base}$ verbs are those that appear after modal verbs like may and can.]

Problem 5: Coordination and Tense

For the most part, the inflectional rules for verbs stand in a one-to-one relationship with FORM values. The exceptions are the 3rd-Singular, Non-3rd-Singular, and Past-Tense Verb Lexical Rules, all of which produce outputs that are $[FORM \ fin]$. The alternative would be to posit a distinct FORM value for each rule: say, ‘$3sg_{present}$’, ‘$non3sg_{present}$’ and ‘past’, or at least two different forms ‘present’ and ‘past’. Making reference to the discussion of FORM and coordination in section 8.5.2, explain why the decision to use just one FORM value (‘fin’) is right or wrong. Be sure to consider examples where finite VPs that differ in tense are coordinated.

Problem 6: Conjoined Conjunctions
A. Does our grammar license the (ungrammatical) string in (i)? (Assume lexical entries for *and*, *but* and *or* that are all [HEAD *conj.*])

(i) Kim left and but and or but and and Sandy stayed.

B. If you answered ‘yes’ to part (A), draw a tree showing a structure that the grammar licenses for the sentence. (Abbreviated node labels are fine.) If you answered ‘no’ to part (A), explain how it is ruled out.

Problem 7: Arguments in Japanese

As noted in Chapter 2, Japanese word order differs from English in a number of ways, including the fact that it is a ‘Subject-Object-Verb’ (SOV) language. Here are a few relevant examples. In the glosses, ‘NOM’, ‘ACC’, and ‘DAT’ stand for nominative, accusative, and dative case, respectively. (Note that Japanese has one more case – dative – than English does. This doesn’t have any important effects on the analysis; it merely requires that we posit one more possible value of CASE for Japanese than for English). 37

(i) *Hitorino otoko-ga sono hon-o yonda*

one man-NOM that book-ACC read-past

‘One man read that book’

[cf. *Yonda hitorino otoko-ga sono hon-o*

*Otoko-ga hitorino sono hon-o yonda*

*Hitorino otoko-ga hon-o sono yonda*

*Hitorino otoko-ni-o sono hon-o yonda*

*Hitorino otoko-ga sono hon-ga/-ni yonda.]*

(ii) *Hanako-ga hon-o yonda*

Hanako-NOM book-ACC read-past

‘Hanako read the book(s)’

[cf. *Yonda Hanako-ga hon-o*

*Hanako-ga yonda hon-o*

*Hanako-ni-o hon-o yonda*

*Hanako-ga hon-ni-ga/-ni yonda.]*

(iii) *sensei-ga Taro-o-ni sono hon-o ageta*

teacher-NOM Taroo-DAT that book-ACC gave-past

‘The teacher(s) gave that book to Taro’

[cf. *Ageta sensei-ga Taro-o-ni sono hon-o*

*Sensei-ga ageta Taroo-ni sono hon-o*

*Sensei-ga Taro-o-ni ageta sono hon-o*]

*Sensei-o-ni Taro-o-ni sono hon-o ageta*  

*Sensei-ga Taro-o/-o sono hon-o ageta*  

*Sensei-ga Taro-o-ni sono hon-ga/-ni ageta.]*

37 The examples marked with ‘*’ here are unacceptable with the indicated meanings. Some of these might be well formed with some other meaning of no direct relevance; others might be well formed with special intonation that we will ignore for present purposes.
As the contrasting ungrammatical examples show, the verb must appear in final position in Japanese. In addition, we see that verbs select for NPs of a particular case, much as in English. In the following tasks, assume that the nouns and verbs of Japanese are inflected words, derived by lexical rule from the appropriate lexemes.

A. Write Head-Specifier and Head-Complement Rules for Japanese that account for the data illustrated here. How are they different (if at all) from the Head-Specifier and Head-Complement Rules for English?

B. Give the lexical entry for each of the verbs illustrated in (i)–(iv). The data given permit you to specify only some features; leave others unspecified. Make sure your entries interact with the rules you formulated in Task 1 to account for the above data.

C. Assume that nouns like Taroo, hon, etc. are entered in the Japanese lexicon as nominal lexemes. Give lexical entries for these two lexemes (again limiting the features specified to those for which you have data).

D. Formulate three lexical rules for deriving the inflected forms (the words ending in -ga, -o, and -ni) from the nominal lexemes.

---

**Problem 8: Japanese Causatives**

Crosslinguistically, causative constructions like (i) can be either *periphrastic* or *morphological*. In a periphrastic causative (such as (i)), a separate word (typically a verb) expresses the causation and licenses or selects for the causer argument. In a morphological causative, such as the Japanese example in (iii), the causation is expressed by an affix and the verb's valence is augmented by one.

(i) Kim made Sandy eat the cake.
(ii) Suzuki-ga keeki-o tabeta
    Suzuki-NOM cake-ACC eat.past
    ‘Suzuki ate the cake.’
(iii) Aoki-ga Suzuki-ni keeki-o tabesaseta
    Aoki-NOM Suzuki-DAT cake-ACC eat.caused.past
    ‘Aoki made Suzuki eat the cake.’
    [cf. *Aoki-ga Suzuki-ni keeki-o tabeta.]

A. What is the case of the CAUSER argument in (iii)?
B. What is the case of the CAUSEE argument in (iii)?
C. Assume that the lexical entry for *tabeta* in (ii) is as in (iv) and that the semantics of the lexical entry for *tabesaseta* in (iii) is as in (v).

---

38 The ‘...' in the RESTR lists indicate that there should be something more in these entries, namely, a representation of the semantics of past tense.
D. Write a Causative Lexical Rule for Japanese that will derive entries like *tabesase* from entries like *tabe*. (*Tabesase* and *tabe* are the stem forms for *tabesaseta* and *tabeta* respectively. That is, they are the forms that are input to the Past-Tense Verb Lexical Rule. Be sure to make your Causative Lexical Rule a derivational rule.)