3

Analyzing Features of Grammatical Categories

3.1 Introduction

In the last chapter, we saw that there are constraints on which words can go together (what linguists call CO-OCCURRENCE RESTRICTIONS) that are not adequately described using the standard formalism of context-free grammar. Some verbs must take an object; others can never take an object; still others (e.g. *put*, *hand*) require both an object and another phrase of a particular kind. These co-occurrence restrictions, as we have seen, give rise to a great deal of redundancy in CFGs. In addition, different forms of a given verb impose different conditions on what kind of NP can precede them (i.e. on what kind of subject they co-occur with). For example, *walks* requires a third-person singular NP as its subject; *walk* requires a plural subject, or else one that is first- or second-person singular. As we saw in the last chapter, if we try to deal with this complex array of data by dividing the category V into more specific categories, each with its unique co-occurrence restrictions, we end up with a massively redundant grammar that fails to capture linguistically significant generalizations.

We also isolated a second defect of CFGs, namely that they allow rules that are arbitrary. Nothing in the theory of CFG reflects the headedness of phrases in human language – that is, the fact that phrases usually share certain key properties (nounhood, verbhood, prepositionhood, etc.) with a particular daughter within them.We must somehow modify the theory of CFG to allow us to express the property of headedness.

Our solution to the problem of redundancy is to make grammatical categories decomposable into component parts. CFG as presented so far treats each grammatical category symbol as ATOMIC – that is, without internal structure. Two categories are either identical or different; there is no mechanism for saying that two categories are alike in some ways, but different in others. However, words and phrases in natural languages typically behave alike in certain respects, but not in others. For example, the two words *deny* and *denies* are alike in requiring an NP object (both being forms of a transitive verb). But they differ in terms of the kind of subject NP they take: *denies* requires a third-person-singular subject like *Kim* or *she*, while *deny* accepts almost any NP subject except the third-person-singular kind. On the other hand, *denies* and *disappears* both take a singular subject NP, but only the former can co-occur with a following object NP. In other words,

the property of taking a third-person-singular subject is independent of the property of taking a direct object NP. This is illustrated in the following table:

(1)

	3rd singular subject	plural subject
direct object NP	*denies*	*deny*
no direct object NP	*disappears*	*disappear*

The table in (1) illustrates only two of the cross-cutting properties of verbs. There are many more. For example, the properties of forming the third-person-singular form with *-s*, the past tense form with *-ed*, and the present participle with *-ing* are all orthogonal to the property of taking a direct object NP. In Chapter 8, we will see how to write rules for generating these INFLECTIONAL forms of verbs. In order to write such rules with maximal generality, we need to be able to refer to the class of all verbs, regardless of whether they take a direct object NP. More generally, an adequate theory of grammar needs to be able to categorize words into classes defined in terms of cross-cutting properties. In Chapter 2, we showed CFG to be inadequate as a theory of grammar, because it provides no means to represent cross-cutting properties. Instead, it ends up proliferating atomic categories and missing generalizations.

To accommodate these observations, we will develop the view that grammatical categories are not atomic, but rather are COMPLEXES of grammatical properties. In some ways, this innovation is similar to the periodic table of the elements in chemistry, which represents the elements as complexes of physical properties. The rows and columns of the table represent classes of elements that have properties in common, and the classes interesect: each element belongs to more than one class, and shares only some of its properties with the other elements in each of the classes it belongs to. Treating grammatical categories as complexes of grammatical properties will also pave the way for a solution to the second defect of CFGs, by allowing us to express the property of headedness.

3.2 Feature Structures

This section introduces the formal mechanism we will use for representing grammatical categories as complexes of grammatical properties. But let us first review the grammatical properties we have covered so far. We have seen that verbs differ in their transitivity. In fact, this kind of variation is not restricted to verbs. More generally, linguists talk about elements that have different combinatoric potential in terms of differing VALENCE.[1] Likewise, we talk of the NUMBER (singular or plural) of a noun, the PART OF SPEECH of a word (whether it's a noun, verb, etc.), and the FORM of a verb (e.g. whether it is a present participle, an infinitive, etc.). Previously we have been associating each word in the lexicon with a single atomic category (such a P, N-SG, etc.). Now, in order to model grammatical categories as complexes of information, we will use FEATURE STRUCTURES instead of atomic labels.

A feature structure is a way of representing grammatical information. Formally, a feature structure consists of a specification of a set of features (which we will write in upper case), each of which is paired with a particular value. Feature structures can be

[1]This term, borrowed from chemistry, refers to the capacity to combine with atoms, ions, and the like.

thought of in at least two roughly equivalent ways. For example, they may be conceived of as functions (in the mathematicians' sense of the word)[2] specifying a value for each of a set of features, or else as directed graphs where feature names label arcs that point to appropriately labeled nodes. For grammatical purposes, however, it will be most useful for us to focus on DESCRIPTIONS of feature structures, which we will write in a square bracket notation, as shown in (2):

(2) $$\begin{bmatrix} \text{FEATURE}_1 & \text{VALUE}_1 \\ \text{FEATURE}_2 & \text{VALUE}_2 \\ \cdots & \\ \text{FEATURE}_n & \text{VALUE}_n \end{bmatrix}$$

For example, we might treat the category of the word *bird* in terms of a feature structure that specifies just its part of speech and number. We may assume such a category includes appropriate specifications for two appropriately named features: its part of speech (POS) is noun, and its number (NUM) is singular (sg). Under these assumptions, the lexical entry for *bird* would be a pair consisting of a form and a feature structure description, roughly as shown in (3):[3]

(3) $$\left\langle \text{bird} , \begin{bmatrix} \text{POS} & \text{noun} \\ \text{NUM} & \text{sg} \end{bmatrix} \right\rangle$$

One of the first things we will want to do in developing a theory of grammar is to classify linguistic entities in various ways. To this end, it is particularly useful to introduce the notion of TYPE. This concept is really quite simple: if we think of a language as a system of linguistic entities (words, phrases, categories, sounds, and other more abstract entities that we will introduce as we go along), then types are just classes of those entities. We assign entities to these classes on the basis of certain properties that they share. Naturally, the properties we employ in our type classification will be those that we wish to refer to in our descriptions of the entities. Thus each grammatical type will be associated with particular features and sometimes with particular values for those features. As we develop our theory of grammatical types, we will in fact be developing a theory of what kinds of linguistic entities there are, and what kinds of generalizations hold of those entities.

Let us make this very abstract discussion more concrete by considering the use of feature structures to elucidate a simple nonlinguistic domain: universities and the people who are associated with them. We'll start from the assumption that the people and the other entities are really 'out there' in the real world. Our first step then in constructing a theory of this part of the world is to develop a model. A simple model will be a set of mathematical entities that we assume to correspond to the real ones. Our theory will be successful to the extent that we can show that the properties that our theory ascribes to our modeling entities (through stipulation or deduction from the stipulations) also hold

[2]A function in this sense is a set of ordered pairs such that no two ordered pairs in the set share the same first element. What this means for feature structures is that each feature in a feature structure must have a unique value.

[3]Throughout this book, we will describe linguistic forms in terms of standard English orthography. In fact, a lexical entry such as this should contain a phonological description that will play a role in the word's phonological realization, a topic we will not consider in detail here.

of the real world entities that they correspond to.

The domain at hand includes entities such as universities, departments and individuals (people). We might want to talk about certain properties of these entities, for example their names or telephone numbers. In order to do so, we will start to build our model by declaring the existence of a general type called *entity* and say that the features NAME and TEL(EPHONE) are appropriate features for all entities (tokens) of this type. So for each university, department, or person in this university world, we would hypothesize a distinct feature structure model that we could describe as follows:

(4) a.
$$\begin{bmatrix} \textit{entity} & \\ \text{NAME} & \text{Stanford University} \\ \text{TEL} & \text{650-723-2300} \end{bmatrix}$$

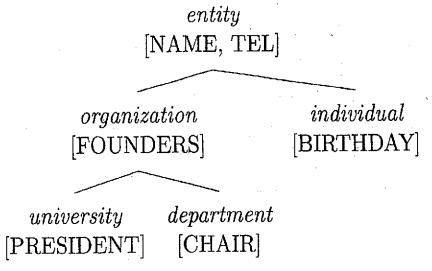
b.
$$\begin{bmatrix} \textit{entity} & \\ \text{NAME} & \text{John Hennessy} \\ \text{TEL} & \text{650-723-2481} \end{bmatrix}$$

c.
$$\begin{bmatrix} \textit{entity} & \\ \text{NAME} & \text{Stanford Linguistics} \\ \text{TEL} & \text{650-723-4284} \end{bmatrix}$$

Note that we use type names (in this case *entity*), written in italics, as labels on the top line within feature structures.

Of course 'entity' is a very general classification – our theory would not have progressed far if it recognized no more specific kinds of things. So in fact, we would want our theory to include the fact that there are different subtypes of the type *entity*. Let's call these new types *university*, *department*, and *individual*. Entities belonging to each of these types have their own special properties. For example, individual people have birthdays, but universities and departments don't (or not in the same sense). Similarly, departments have chairs (or 'heads of department'), but neither universities nor individuals do. And only universities have presidents. Finally, universities and deparments, but not individuals, have founders, a fact that will motivate grouping these two types together under a common intermediate-level type which we will call *organization*. We can then accommodate all these facts by declaring each of the relevant features (BIRTHDAY, CHAIR, PRESIDENT, FOUNDERS) to be appropriate for entities of the appropriate subtype. This organization of the types of entity and the features that are appropriate for each of them results in the TYPE HIERARCHY shown in (5):

(5)



Each type of entity has its own constellation of features – some of them are declared appropriate for entities of the indicated type; others are sanctioned by one of the supertypes: *entity* or *organization*. This is a simple illustration of how a hierarchical classification system works. A given feature structure contains only those features that are declared appropriate by one of its types, that is, by its LEAF type[4] or one of its supertypes in a hierararchy like (5). This formal declaration is just a precise way of saying that the members of the relevant subclasses have certain properties that distinguish them from other entities in the system, as well as certain properties that they share with other entities.

Now that we've extended the model by adding types and features, the resulting descriptions that we write will be appropriately more specific, as in (6):

(6) a.
$$\begin{bmatrix} \textit{university} & \\ \text{NAME} & \text{Stanford University} \\ \text{FOUNDERS} & \langle \text{Leland Stanford , Jane Stanford} \rangle \\ \text{PRESIDENT} & \text{John Hennessy} \\ \text{TEL} & \text{650-723-2300} \end{bmatrix}$$

b.
$$\begin{bmatrix} \textit{individual} & \\ \text{NAME} & \text{John Hennessy} \\ \text{BIRTHDAY} & \text{9-22-52} \\ \text{TEL} & \text{650-723-2481} \end{bmatrix}$$

c.
$$\begin{bmatrix} \textit{department} & \\ \text{NAME} & \text{Stanford Linguistics} \\ \text{FOUNDERS} & \langle \text{Joseph Greenberg , Charles Ferguson} \rangle \\ \text{CHAIR} & \text{Eve Clark} \\ \text{TEL} & \text{650-723-4284} \end{bmatrix}$$

Note that we also need to specify what kind of value is appropriate for each feature. Here we've used angled brackets ('⟨' and '⟩') to construct a list as the value of the feature FOUNDERS. As we will see, a feature structure also inherits any type constraints, (that is, potentially complex constraints on feature values) that are associated with its supertypes. Articulating a type hierarchy and the constraints associated with each type in the hierarchy is an important component of a theory that uses typed feature structures as its models.

Let us reconsider the feature structures in (6). These structures, as explicated above, aren't yet expressing the proper information about the objects they are trying to model. In particular, the value of features like PRESIDENT and CHAIR are atomic, i.e. the names John Hennessy and Eve Clark. But this isn't right – the president of Stanford University is the individual John Hennessy, not his name. The same goes for Eve Clark, who gives more to being chair of the Stanford Linguistics Department than just her name.

[4] The leaf types are the basic or bottom-level types in a hierarchy, i.e. the types that have no subtypes. These are also referred to in the literature (somewhat counterintuitively) as 'maximal' types.

Similarly, the value of the FOUNDERS feature should be a list of individuals, not a list of names. To reflect these observations, we now introduce complex feature structures, those whose features may have nonatomic feature structures (or lists of feature structures) as their value, where appropriate. This modification leads to the following more accurate models of Stanford and its Linguistics Department (the model of John Hennessy remains unaffected by this change):

(7) a.
$$\begin{bmatrix} \textit{university} & \\ \text{NAME} & \text{Stanford University} \\ \text{FOUNDERS} & \left\langle \begin{bmatrix} \textit{individual} \\ \text{NAME} \quad \text{Leland Stanford} \end{bmatrix}, \begin{bmatrix} \textit{individual} \\ \text{NAME} \quad \text{Jane Stanford} \end{bmatrix} \right\rangle \\ \text{PRESIDENT} & \begin{bmatrix} \textit{individual} & \\ \text{NAME} & \text{John Hennessy} \\ \text{BIRTHDAY} & \text{9-22-52} \\ \text{TEL} & \text{650-723-2481} \end{bmatrix} \\ \text{TEL} & \text{650-723-2300} \end{bmatrix}$$

b.
$$\begin{bmatrix} \textit{department} & \\ \text{NAME} & \text{Stanford Linguistics} \\ \text{FOUNDERS} & \left\langle \begin{bmatrix} \textit{individual} & \\ \text{NAME} & \text{Joseph Greenberg} \\ \text{BIRTHDAY} & \text{5-28-15} \end{bmatrix}, \begin{bmatrix} \textit{individual} & \\ \text{NAME} & \text{Charles Ferguson} \\ \text{BIRTHDAY} & \text{7-6-21} \end{bmatrix} \right\rangle \\ \text{CHAIR} & \begin{bmatrix} \textit{individual} & \\ \text{NAME} & \text{Eve Clark} \\ \text{BIRTHDAY} & \text{7-26-42} \\ \text{TEL} & \text{650-723-4284} \end{bmatrix} \\ \text{TEL} & \text{650-723-4284} \end{bmatrix}$$

When we model some empirical problem in this way, it is important to distinguish the modeling objects (the typed feature structures) from the statements we make about them. The objects in our model are meant to be simplified analogs of objects in the real world (if they weren't simplified, it wouldn't be a model). The statements we make about the modeling objects – our constraints – constitute our theory of the domain we are investigating. The system of types we set up of course is the first step in developing such a theory:

- It states what kinds of objects we claim exist (the types).
- It organizes the objects hierarchically into classes with shared properties (the type hierarchy).
- It states what general properties each kind of object has (the feature and feature value declarations).

We could summarize the beginnings of our theory of universities in terms of the following table (where 'IST' stands for 'immediate supertype'):[5]

(8)

TYPE	FEATURES/VALUES	IST
entity	[NAME *string* TEL *number*]	
organization	[FOUNDERS *list(individual)*]	*entity*
university	[PRESIDENT *individual*]	*organization*
department	[CHAIR *individual*]	*organization*
individual	[BIRTHDAY *date*]	*entity*

Against this background, it is the particular constraints we write that fill in the details. Type constraints specify properties that relevant classes of objects have (e.g. that universities have presidents who are individuals) and other constraints characterize properties of certain idiosyncratic entities that we find it necessary to recognize (e.g. that Stanford's president is John Hennessy). We then make the standard assumption that our modeling objects are in correspondence with the real world. In so doing, our constraints are making claims about reality in ways that distinguish our theory of the relevant empirical domain from many others that could be formulated.

Our (admittedly somewhat artificial) theory of Stanford University then consists of a set of constraints that reflect our claims about the way Stanford is, some of which may reflect the way all universities are. Those constraints are meant to describe (or be SATISFIED by) the objects in our model of Stanford – the feature structures assigned to appropriate types, exhibiting the relevant properties. And if we've modeled things correctly, our feature structures will reflect the reality of Stanford and we will view our theory as making correct predictions.

Theories often include constraints requiring two things to be identical. For example, suppose we wanted to state the hypothesis that the phone number of a department chair was always the same as the department's phone number. This somewhat trivial (yet precise) claim might be formulated as follows:

[5] Note that this table assumes the types *number*, *string* and *date*. These three types would also need to be incorporated into the type hierarchy.

(9) $$\textit{department}: \begin{bmatrix} \text{TEL} & \boxed{1} \\ \text{CHAIR} & \begin{bmatrix} \text{TEL} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

The colon here denotes a conditional ('if–then') relation between a type and a claim being made about the instances of that type. The boxed numerals in (9) are called 'tags'. They function like variables in algebra, logic, or programming languages. That is, they indicate that two values within a given feature structure are identical. What the constraint in (9) is saying then is that for any feature structure of type *department*, if you start at the outside and follow the feature path CHAIR|TEL, you'll arrive at the same value that you find when you start at the outside again and follow the (single-feature) path TEL.

Of course, it's easy to test the predictions of a one-sentence theory like (9). The feature structure models of type *department* that satisfy (9) have a clear and simple property and the relevant objects out in the real world are all listed in the Stanford Directory with their phone numbers. It's presumably not hard to verify whether (9) is true or not.[6] But science is full of theories whose predictions are much harder to test. Indeed, we'll see that evaluating the predictions of a theory of language based on feature structure models can sometimes be quite a subtle matter.

Interesting theories involve a number of different claims that interact. For this reason, it's essential that we have a way of combining constraints and determining which models satisfy the resulting combinations, however complex they might be. We will in fact use a simple method for combining (conjoining) constraints – one that we'll sometimes write with the symbol '&', as in (10a). Quite often, however, we will simply combine two constraints into a bigger one like (10b):[7]

(10) a. $\begin{bmatrix} \text{TEL} & \text{650-723-4284} \end{bmatrix}$ & $\begin{bmatrix} \text{NAME} & \text{Stanford Linguistics} \end{bmatrix}$

b. $\begin{bmatrix} \text{NAME} & \text{Stanford Linguistics} \\ \text{TEL} & \text{650-723-4284} \end{bmatrix}$

Notice how our constraints relate to their models. The first conjunct (the bracketed constraint before the '&' in (10a)) is satisfied by a set of feature structures (in our current model, it's the set that contains the feature structure we used to model the Stanford Linguistics Department and the one we used to model its chair). The second conjunct in (10a) is also satisfied by a set of feature structures, but this set has only one member: the feature structure serving as our model of the Stanford Linguistics Department. And the constraint in (10), whether we formulate it as in (10a) or as in (10b), is satisfied by the intersection of the two other sets, i.e. by the (singleton) set that contains just the feature structure we used to model the Stanford Linguistics Department.

[6] In fact, this theory of Stanford department phone numbers is easily falsified.

[7] The process of combining constraints in the fashion of (10b) is often called 'unification'. Theories of the sort we describe in this book are sometimes called 'unification-based', but this term is misleading. Unification is a method (i.e. a procedure) for solving sets of identity constraints. But it is the constraints themselves that constitute the theory, not any procedure we might use with them. Hence, we will refer to the theory of grammar we devlop, and the class of related theories, as 'constraint-based', rather than 'unification-based'.

Note that the constraints in (11) are incompatible because they differ in the value they assign to the feature NAME:

(11) a. $\begin{bmatrix} \textit{university} \\ \text{NAME} & \text{Stanford University} \end{bmatrix}$

b. $\begin{bmatrix} \textit{university} \\ \text{NAME} & \text{Harvard University} \end{bmatrix}$

And because (11a) and (11b) are incompatible, they couldn't be used to describe the same entity.

Similarly, the constraints in (12) cannot be combined:

(12) a. $\begin{bmatrix} \textit{individual} \\ \text{TEL} & \text{650-555-4284} \end{bmatrix}$

b. $\begin{bmatrix} \textit{department} \\ \text{TEL} & \text{650-555-4284} \end{bmatrix}$

In this case, the problem is that (12a) and (12b) specify incompatible types, namely, *individual* and *department*. Hence (12a) and (12b) must be describing distinct entities. But the constraint in (13) is compatible with any of those in (14a)–(14c):

(13) $\begin{bmatrix} \text{TEL} & \text{888-234-5789} \end{bmatrix}$

(14) a. $\begin{bmatrix} \textit{university} \end{bmatrix}$

b. $\begin{bmatrix} \textit{individual} \\ \text{NAME} & \text{Sailor Moon} \end{bmatrix}$

c. $\begin{bmatrix} \textit{department} \\ \text{NAME} & \text{Metaphysics} \\ \text{CHAIR} & \text{Alexius Meinong, Jr.} \end{bmatrix}$

For example, the combination of (13) and (14b), shown in (15), is satisfied by those objects (in our model) that satisfy both (13) and (14b):

(15) $\begin{bmatrix} \textit{individual} \\ \text{NAME} & \text{Sailor Moon} \\ \text{TEL} & \text{888-234-5789} \end{bmatrix}$

Finally, the constraints in (16) cannot be combined:

(16) a. $\begin{bmatrix} \text{BIRTHDAY} & \text{10-10-1973} \end{bmatrix}$

b. $\begin{bmatrix} \text{PRESIDENT} & \begin{bmatrix} \textit{individual} \\ \text{NAME} & \text{Sailor Moon} \end{bmatrix} \end{bmatrix}$

In this case, the constraints cannot be combined because there is no type for which the features BIRTHDAY and PRESIDENT are appropriate. Since all of the modeling objects must belong to some type, there will be none that satisfy both (16a) and (16b).

When our feature structure constraints get a bit more complicated, we will sometimes want to indicate simultaneously the value of a particular feature and the fact that that value is identical with the value of another feature (or feature path), as shown in (17):

(17) $\begin{bmatrix} \text{TEL} & \boxed{1}650\text{-}723\text{-}4284 \\ \text{CHAIR} & [\text{TEL} \quad \boxed{1}] \end{bmatrix}$

But it would make no difference if we wrote the phone number after the other occurrence of $\boxed{1}$ in (17):

(18) $\begin{bmatrix} \text{TEL} & \boxed{1} \\ \text{CHAIR} & [\text{TEL} \quad \boxed{1}650\text{-}723\text{-}4284] \end{bmatrix}$

The intended interpretation would be exactly the same. It also makes no difference what order we write the features in. For example, (17) and (18) are both equivalent to either of the following:

(19) a. $\begin{bmatrix} \text{CHAIR} & [\text{TEL} \quad \boxed{1}650\text{-}723\text{-}4284] \\ \text{TEL} & \boxed{1} \end{bmatrix}$

b. $\begin{bmatrix} \text{CHAIR} & [\text{TEL} \quad \boxed{1}] \\ \text{TEL} & \boxed{1}650\text{-}723\text{-}4284 \end{bmatrix}$

Finally, it should be noticed that the choice of a particular tag is also completely arbitrary. The following constraints are also equivalent to the ones in (17)–(19):

(20) a. $\begin{bmatrix} \text{CHAIR} & [\text{TEL} \quad \boxed{279}650\text{-}723\text{-}4284] \\ \text{TEL} & \boxed{279} \end{bmatrix}$

b. $\begin{bmatrix} \text{TEL} & \boxed{\text{N}} \\ \text{CHAIR} & [\text{TEL} \quad \boxed{\text{N}}650\text{-}723\text{-}4284] \end{bmatrix}$

These are still simple examples. In the chapters that follow, we will have occasion to combine the various tools introduced here into fairly complex constraints.

Exercise 1: Practice with Combining Constraints

Are the following pairs of constraints compatible? If so, what does the combined constraint look like?

A. $\begin{bmatrix} \text{TEL} & 650\text{-}723\text{-}4284 \end{bmatrix} \& \begin{bmatrix} \textit{department} \\ \text{NAME} \quad \text{Metaphysics} \end{bmatrix}$

B. $\begin{bmatrix} \text{TEL} & 650\text{-}723\text{-}4284 \end{bmatrix} \& \begin{bmatrix} \text{TEL} & \boxed{23} \\ \text{CHAIR} & [\text{TEL} \ \boxed{23}] \end{bmatrix}$

C. $\begin{bmatrix} \text{PRESIDENT} & \boxed{1} \\ \text{FOUNDERS} & \langle \boxed{1} \rangle \end{bmatrix} \& \begin{bmatrix} \textit{individual} \\ \text{NAME} \quad \text{John Hennessy} \end{bmatrix}$

3.3 The Linguistic Application of Feature Structures

3.3.1 Feature Structure Categories

So how do typed feature structures help us with our linguistic concerns? Instead of saying that there is just one kind of linguistic entity, which must bear a value for every feature we recognize in our feature structures, we will often want to say that a given entity is of a certain type for which only certain features are appropriate. In fact, we will use typing in many ways: for example, to ensure that [NUM sg] (or [NUM pl]) can only be specified for certain kinds of words (for example, nouns, pronouns, and verbs), but not for prepositions or adjectives.[8] Likewise, we will eventually introduce a feature AUX to distinguish auxiliaries (helping verbs like *will* and *have*) from all other verbs, but we won't want to say that nouns are all redundantly specified as [AUX –]. Rather, the idea that we'll want our grammar to incorporate is that the feature AUX just isn't appropriate for nouns. We can use types as a basis for classifying the feature structures we introduce and the constraints we place on them. In so doing, we provide an easy way of saying that particular features only go with certain types of feature structure. This amounts to the beginnings of a linguistic ontology: the types lay out what kinds of linguistic entities exist in our theory, and the features associated with those types tell us what general properties each kind of entity exhibits.[9]

In addition, the organization of linguistic objects in terms of a type hierarchy with intermediate types (analogous to *organization* in the university example) is significant. Partial generalizations – generalizations that hold of many but not all entities – are very common in the domain of natural language. Intermediate types allow us to state those generalizations. This feature of our theory will become particularly prominent when we organize the lexical entries into a hierarchy in Chapter 8.

In this chapter, we will develop a feature-based grammar that incorporates key ideas from the CFG we used in Chapter 2. We will show how feature structures can solve some of the problems we raised in our critical discussion of that grammar. As we do so, we will gradually replace all the atomic category names used in the CFG (S, NP, V, etc.) by typed feature structures. Since the grammar presented in this chapter is modeled on the CFG of Chapter 2, it is just an intermediate step in our exposition. In Chapter 4, we will refine the Chapter 3 grammar so that in the chapters to come we can systematically expand its coverage to include a much wider set of data.

3.3.2 Words and Phrases

To start with, let us draw a very intuitive distinction between two types: *word* and *phrase*. Our grammar rules (i.e. our phrase structure rules) all specify the properties of phrases;

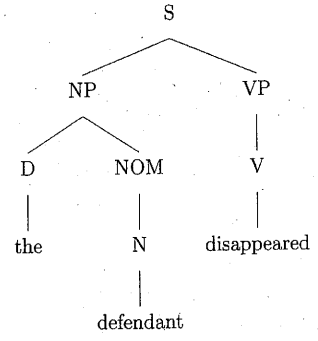
[8] Many such restrictions are language-particular. For example, adjectives are distinguished according to number (agreeing with the noun they modify) in many languages. Even prepositions exhibit agreement inflection in some languages (e.g. modern Irish) and need to be classified in similar terms.

[9] We might instead introduce some mechanism for directly stipulating dependencies between values of different features – such as a statement that the existence of a value for AUX implies that the value of POS is 'verb'. (For a theory that incorporates a mechanism like this, see Gazdar et al. 1985.) But mechanisms of this kind are unnecessary, given the availability of types in our theory.

the lexicon provides a theory of words.

Consider the CFG tree in (21):

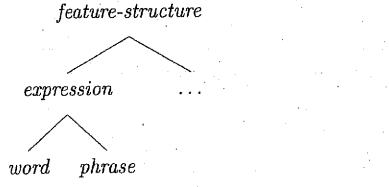
(21)



In this tree, the nodes S, NP, NOM, and VP are all *phrases*. The nodes D, N and V are all *words*. Both of these statements may seem unintuitive at first, because the words *word* and *phrase* are used in various ways. Sometimes a particular form, e.g. *the*, *defendant* or *disappeared*, is referred to as a word and certain sequences of forms, e.g. *the defendant* are called phrases. In the sense we intend here, however, 'word' refers to the category that the lexicon associates with a given form like *disappeared* and 'phrase' refers to the category that the grammar associates with a sequence of such forms.

Although there is an intuitive contrast between *words* and *phrases*, they also have some properties in common, especially in contrast to the more abstract grammatical types we will be positing below. We will therefore create our type hierarchy so that *word* and *phrase* are both subtypes of *expression*:[10]

(22)



One property that words and phrases have in common is part of speech. In the CFG of Chapter 2, this similarity was represented mnemonically (although not formally) in the atomic labels we choose for the categories: NP and N have in common that they are essentially nominal, VP and V that they are essentially verbal, etc. With feature structures, we can represent this formally. We will assume that all *expressions* specify values for a feature we will call HEAD. The value of HEAD will indicate the expression's part of speech. This feature is called HEAD because the part of speech of a phrase depends on the part of speech of one particular daughter, called the head daughter. That is, an NP structure is nominal because it has an N inside of it. That N is the head daughter of the NP structure.

[10]Note that the most general type in our theory will be called *feature-structure*. All of the types we introduce will be subtypes of *feature-structure*. If we were to fully flesh out the university example, something similar would have to be done there.

So far, then, our feature structure representation of the category NP looks like this:

(23) $\begin{bmatrix} \textit{phrase} \\ \text{HEAD} \quad \textit{noun} \end{bmatrix}$

and our feature structure representation of the lexical entry for a noun, say *bird*, looks like this:

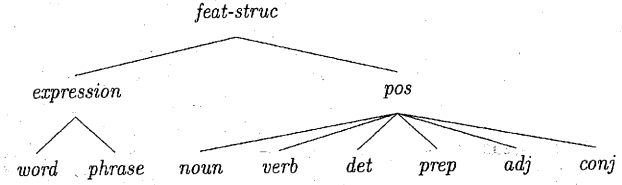
(24) $\left\langle \text{bird} , \begin{bmatrix} \textit{word} \\ \text{HEAD} \quad \textit{noun} \end{bmatrix} \right\rangle$

3.3.3 Parts of Speech

Let us reflect for a moment on parts of speech. There are certain features that are appropriate for certain parts of speech, but not others. We proposed above to distinguish helping verbs from all others in terms of the feature AUX(ILIARY), which will be appropriate only for verbs. Likewise, we will use the feature AGR(EEMENT) only for nouns, verbs, and determiners. To guarantee that only the right features go with the right parts of speech, we will introduce a set of types. Then we can declare feature appropriateness for each part of speech type, just as we did in our type hierarchy for Stanford University.

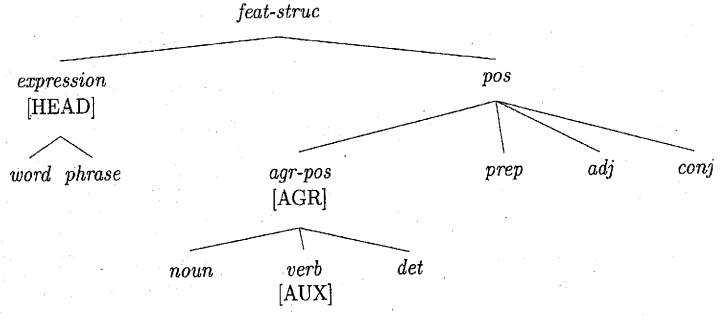
We therefore introduce the types *noun, verb, adj, prep, det*, and *conj* for the six lexical categories we have so far considered. We then make all of these subtypes of a type called *part-of-speech* (*pos*), which is itself a subtype of *feat(ure)-struc(ture)*. The resulting type organization is as shown in (25):

(25)



But in fact, if we want to introduce features only once in a given type hierarchy, then we will have to modify this picture slightly. That's because there are three parts of speech that take the feature AGR.[11] We will thus modify the type hierarchy to give these three types a common supertype where the feature AGR is introduced, as shown in (26):

(26)



[11]There will be a few more as we expand the coverage of our grammar in later chapters.

In this way, determiners and nouns will both specify values for AGR and verbs will specify values for both AGR and AUX. Notice, however, that it is not the words themselves that specify values for these features – rather, it is the feature structures of type *noun*, *verb* or *det*. Individual words (and phrases) get associated with this information because they have a feature HEAD whose value is always a feature structure that belongs to some subtype of *pos*.

So far, we have motivated distinguishing the different subtypes of *pos* as a way of making sure that words only bear features that are appropriate for their part of speech. There is, however, another benefit. As discussed in Section 3.3.2 above, the value of the HEAD feature represents information that a phrase (more precisely, the mother nodes of a phrase structure) shares with its head daughter. (We will see how the grammar enforces this identity in Section 3.3.5 below.) The features we posit for the *pos* types (so far, AGR and AUX) also encode information that phrases share with their head daughters. This is particularly clear in the case of agreement: just as an NP is only nominal because it has an N inside of it, a singular NP is only singular because it has a singular N inside of it. By making AGR a feature of (the relevant subtypes of) *pos*, we can represent this very efficiently: we identify the HEAD value of the mother (say, NP) and that of its head daughter (N). In doing so, we identify not only the mother and head daughter's part of speech, but also any other associated information, for example, their number.[12] In refining our account of the feature structures of type *pos*, we will thus be formulating a general theory of what features the head daughter shares with its mother in a headed phrase.

3.3.4 Valence Features

The approach we are developing also provides a more satisfying analysis of our earlier categories IV, TV, and DTV. Instead of treating these as unanalyzable (i.e. as atoms), we now decompose these as feature structures. To do this, we introduce a new feature VAL (for 'valence'). The value of VAL is a feature structure (of type *val-cat*) representing the combinatoric potential of the word or phrase. The first feature we will posit under VAL is COMPS (for 'complements' – see Chapter 2, Section 2.7), which we use to indicate what the required following environment is for each type of verb: (For now, we assume that the possible values of COMPS are itr = intransitive, str = strict-transitive, and dtr = ditransitive, though we will revise this in the next chapter.)

(27)

$$\text{IV} = \begin{bmatrix} \textit{word} \\ \text{HEAD} & \textit{verb} \\ \text{VAL} & \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} & \text{itr} \end{bmatrix} \end{bmatrix} \qquad \text{TV} = \begin{bmatrix} \textit{word} \\ \text{HEAD} & \textit{verb} \\ \text{VAL} & \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} & \text{str} \end{bmatrix} \end{bmatrix}$$

$$\text{DTV} = \begin{bmatrix} \textit{word} \\ \text{HEAD} & \textit{verb} \\ \text{VAL} & \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} & \text{dtr} \end{bmatrix} \end{bmatrix}$$

[12] We will return to the feature AGR and describe what kinds of things it takes as its value in Section 3.3.6 below.

The three categories described in (27) all share the type *word* and the feature specification [HEAD *verb*]. This is just the combination of types and features that we would naturally identify with the category V. And by analyzing categories in terms of types and features, we can distinguish between the different valence possibilities for verbs, while still recognizing that all verbs fall under a general category. The general category V is obtained by leaving the value of the VAL feature unspecified, as in (28):

(28) $$\text{V} = \begin{bmatrix} \textit{word} \\ \text{HEAD} & \textit{verb} \end{bmatrix}$$

The term UNDERSPECIFICATION is commonly used in linguistics to indicate a less specific linguistic description. Given our modeling assumptions, underspecification has a precise interpretation: an underspecified description (or constraint) always picks out a larger class of feature structures than a fully specified one. In general, the less information given in a description (i.e. the more underspecified it is), the more models (feature structures) there are that will satisfy that description.

In the grammar so far, the category VP differs from the category V only with respect to its type assignment.[13] So VP is recast as the following description:

(29) $$\text{VP} = \begin{bmatrix} \textit{phrase} \\ \text{HEAD} & \textit{verb} \end{bmatrix}$$

And the class of grammatical categories that includes just verbs and verb phrases is defined precisely by the underspecification in (30):

(30) $$\begin{bmatrix} \text{HEAD} & \textit{verb} \end{bmatrix}$$

Similarly, we can reanalyze the categories N and NP as follows:

(31)

$$\text{N} = \begin{bmatrix} \textit{word} \\ \text{HEAD} & \textit{noun} \end{bmatrix} \qquad \text{NP} = \begin{bmatrix} \textit{phrase} \\ \text{HEAD} & \textit{noun} \end{bmatrix}$$

Within this general approach, we can retain all our previous categories (V, S, NP, etc.) as convenient abbreviations.

Underspecification allows us to provide compact descriptions for the sets of categories that our grammar will actually need to refer to, what linguists usually call 'natural classes'. For example, while we couldn't even talk about IV, DTV, and TV as one class in CFG, we can now refer to them together as *words* that are [HEAD *verb*]. We will use the symbol V as an abbreviation for this feature structure description, but it should now be regarded merely as an abbreviated description of the class of typed feature structures just described. The same is true for N, NP, VP, etc.

Observe that the feature analysis we have just sketched does not yet accommodate the category NOM. NP and NOM are both [HEAD *noun*]. And since the COMPS value is used to indicate what the following environment must be, it is not appropriate for the distinction between NP and NOM. Recall that NOM differs from NP in that it

[13] Additional differences with respect to their VAL values will be discussed shortly. A more sweeping reanalysis of the feature composition of these categories is introduced in Chapter 4 and carried on to subsequent chapters.

does not include the determiner, which is at the beginning of the phrase. In fact, it is a straightforward matter to use features to model our three-level distinction among N, NOM, and NP. NOM is the category that includes everything in the NP except the determiner, e.g. *picture of Yosemite* in *that picture of Yosemite*. We can distinguish NOM and NP using features in much the same way that we distinguished transitive and intransitive verbs – that is, by introducing a valence feature that indicates a restriction on the possible contexts in which the category in question can appear. In this case, the feature will specify whether or not a determiner is needed. We call this feature SPR (SPECIFIER). Just as we introduced 'complement' as a generalization of the notion of object, we are now introducing 'specifier' as a generalization of the notion of determiner.

For now, we will treat SPR as having two values: [SPR –] categories need a specifier on their left; [SPR +] categories do not, either because they label structures that already contain a specifier or that just don't need one. Note that like COMPS, SPR encodes an aspect of an expression's combinatoric potential. NP and NOM are thus defined as in (32):

(32)
$$\text{NP} = \begin{bmatrix} \textit{phrase} \\ \text{HEAD} \quad \textit{noun} \\ \text{VAL} \quad \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} \quad \text{itr} \\ \text{SPR} \quad + \end{bmatrix} \end{bmatrix} \qquad \text{NOM} = \begin{bmatrix} \textit{phrase} \\ \text{HEAD} \quad \textit{noun} \\ \text{VAL} \quad \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} \quad \text{itr} \\ \text{SPR} \quad - \end{bmatrix} \end{bmatrix}$$

We can also use the feature SPR to distinguish between VP and S, by treating a subject NP as the VP's specifier. That is, VP and S can be distinguished as follows:

(33)
$$\text{S} = \begin{bmatrix} \textit{phrase} \\ \text{HEAD} \quad \textit{verb} \\ \text{VAL} \quad \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} \quad \text{itr} \\ \text{SPR} \quad + \end{bmatrix} \end{bmatrix} \qquad \text{VP} = \begin{bmatrix} \textit{phrase} \\ \text{HEAD} \quad \textit{verb} \\ \text{VAL} \quad \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} \quad \text{itr} \\ \text{SPR} \quad - \end{bmatrix} \end{bmatrix}$$

In calling both determiners and subject NPs specifiers, we are claiming that the relationship between subject and VP is in important respects parallel to the relationship between determiner and NOM. The intuition behind this claim is that specifiers (subject NPs and determiners) serve to complete the phrases they are in. S and NP are fully formed categories, while NOM and VP are still incomplete. The idea that subjects and determiners play parallel roles seems particularly intuitive when we consider examples like (34).

(34) a. We created a monster.
b. our creation of a monster

We will have more to say about the feature SPR in the next chapter.

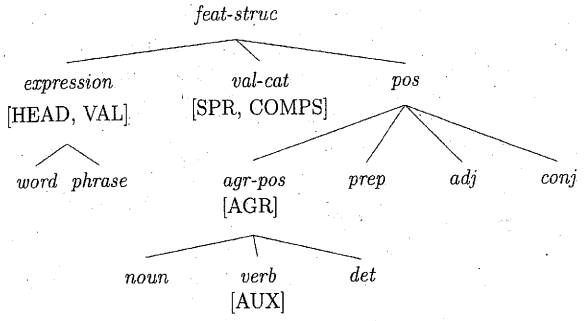
Returning to (32), notice that we have extended the intuitive meaning of the specification [COMPS itr] so that it applies to phrases as well as to words. This is a natural

extension, as phrases (whether NP, S, VP or NOM) are like strictly intransitive verbs in that they cannot combine with complements. (Recall that a phrase contains its head's complement(s), so it can't combine with any more). Notice also that under this conception, the abbreviations NP and S both include the following feature specifications:

(35)
$$\begin{bmatrix} \text{VAL} \quad \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} \quad \text{itr} \\ \text{SPR} \quad + \end{bmatrix} \end{bmatrix}$$

As *words* and *phrases* both need to be specified for the valence features, we declare VAL to be appropriate for the type *expression*. The value of VAL is a *val-cat*, and COMPS and SPR are both features of *val-cat*.[14] Our type hierarchy now looks like this:

(36)



3.3.5 Reformulating the Grammar Rules

Turning now to the phrase structure rules considered in Chapter 2, we can reformulate our VP rules in terms of our new feature structure categories. Consider the following way of stating these rules:

(37) a.
$$\begin{bmatrix} \textit{phrase} \\ \text{HEAD} \quad \boxed{1} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} \quad \text{itr} \\ \text{SPR} \quad - \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textit{word} \\ \text{HEAD} \quad \boxed{1} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} \quad \text{itr} \\ \text{SPR} \quad - \end{bmatrix} \end{bmatrix}$$

b.
$$\begin{bmatrix} \textit{phrase} \\ \text{HEAD} \quad \boxed{1} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} \quad \text{itr} \\ \text{SPR} \quad - \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textit{word} \\ \text{HEAD} \quad \boxed{1} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} \quad \text{str} \\ \text{SPR} \quad - \end{bmatrix} \end{bmatrix} \quad \text{NP}$$

c.
$$\begin{bmatrix} \textit{phrase} \\ \text{HEAD} \quad \boxed{1} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} \quad \text{itr} \\ \text{SPR} \quad - \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textit{word} \\ \text{HEAD} \quad \boxed{1} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} \quad \text{dtr} \\ \text{SPR} \quad - \end{bmatrix} \end{bmatrix} \quad \text{NP NP}$$

[14] In Chapter 5, we will add a further feature, MOD, to *val-cat*.

The two occurrences of [1] in each of these rules tell us that the HEAD value of the mother and that of the first daughter must be identified. Since the rules in (37) were introduced as VP rules, the obvious value to assign to [1] is *verb*. But, by stating the rules in this underspecified way, we can use them to cover some other structures as well. The first rule, for intransitives, can be used to introduce nouns, which can never take NP complements (in English). This is done simply by instantiating [1] as *noun*, which will in turn cause the mother to be a NOM. To make this work right, we will have to specify that lexical nouns, like intransitive verbs, must be [COMPS itr]:

(38)
$$\left\langle \text{bird} , \begin{bmatrix} \textit{word} \\ \text{HEAD} & \textit{noun} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \right\rangle$$

Note that both verbs and nouns are lexically specified as [SPR −], i.e. as having not (yet) combined with a specifier.

We can now recast the CFG rules in (39):

(39) a. S → NP VP
b. NP → (D) NOM

Assuming, as we did above, that S is related to VP and V in just the same way that NP is related to NOM and N, the rules in (39) may be reformulated as (40a) and (40b), respectively:

(40) a.
$$\begin{bmatrix} \textit{phrase} \\ \text{HEAD} & \boxed{1}\textit{verb} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \rightarrow \text{NP} \begin{bmatrix} \textit{phrase} \\ \text{HEAD} & \boxed{1} \\ \text{VAL} & \begin{bmatrix} \text{SPR} & - \end{bmatrix} \end{bmatrix}$$

b.
$$\begin{bmatrix} \textit{phrase} \\ \text{HEAD} & \boxed{1}\textit{noun} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \rightarrow \text{D} \begin{bmatrix} \textit{phrase} \\ \text{HEAD} & \boxed{1} \\ \text{VAL} & \begin{bmatrix} \text{SPR} & - \end{bmatrix} \end{bmatrix}$$

In these rules, 'NP' and 'D' are abbreviations for feature structure descriptions. NP was defined in (32) above. We'll assume that 'D' is interpreted as follows:

(41)
$$\text{D} = \begin{bmatrix} \textit{word} \\ \text{HEAD} & \textit{det} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix}$$

Note that the feature structure rule in (40b) differs from the CFG NP rule in (39b) in that the former makes the determiner obligatory. In fact, the optionality in the CFG rule caused it to overgenerate: while some nouns (like *information* or *facts*) can appear with

or without a determiner, others (like *fact*) require a determiner, and still others (like *you* or *Alex*) never take a determiner:

(42) a. I have the information.
b. I have information.
c. I was already aware of that fact.
d.*I was already aware of fact.
e. I know you.
f.*I know the you.

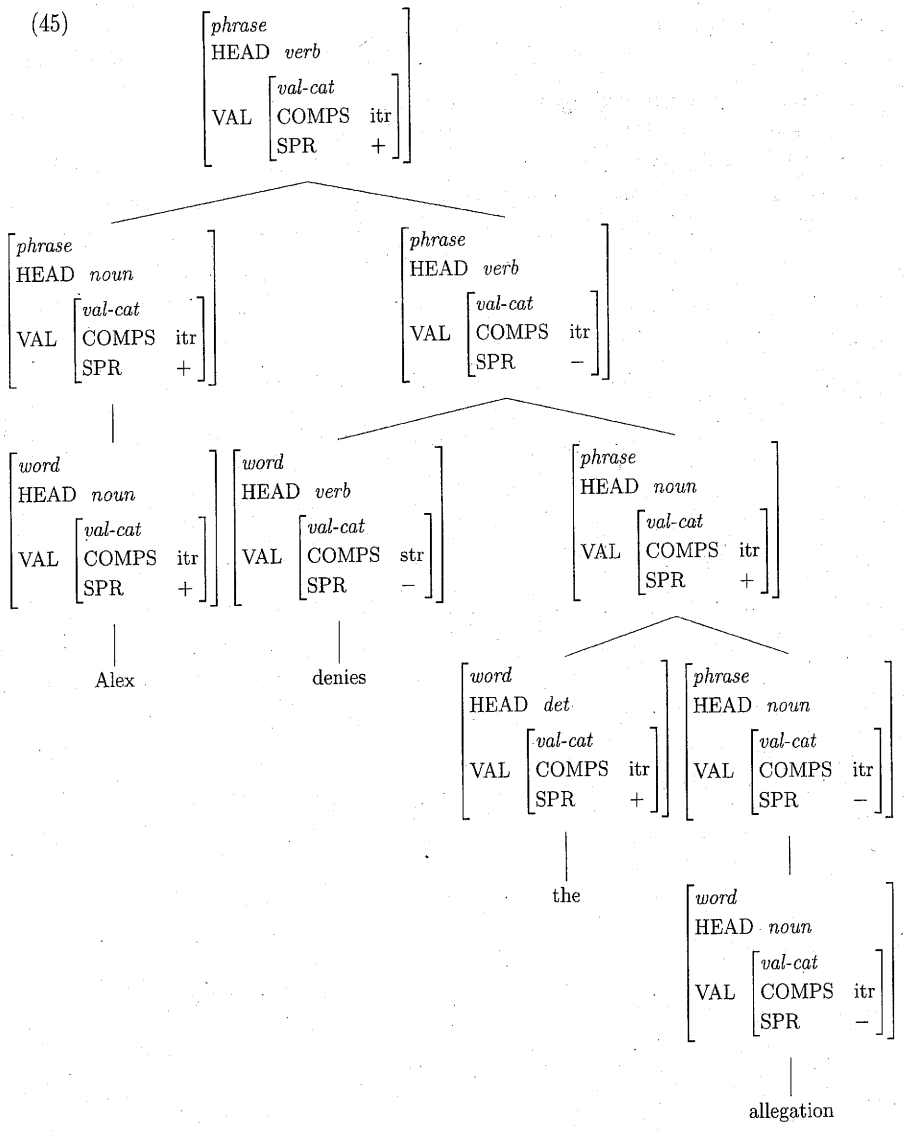
Since the CFG rule in (39b) doesn't distinguish between different kinds of Ns, it in fact licenses all of the NPs in (42). We will return to the problem of nouns whose determiners are truly optional (like *information*) in Chapter 8. The thing to note here is that the feature SPR allows us to distinguish nouns that require determiners (like *fact* or *bird*) from those that refuse determiners (like *you* or *Alex*). The former are specified as [SPR −], and build NPs with the rule in (40b). The latter are [SPR +] (see (43)), and require a new rule, given in (44):

(43)
$$\left\langle \text{Alex} , \begin{bmatrix} \textit{word} \\ \text{HEAD} & \textit{noun} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \right\rangle$$

(44)
$$\begin{bmatrix} \textit{phrase} \\ \text{HEAD} & \boxed{1}\textit{noun} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textit{word} \\ \text{HEAD} & \boxed{1} \\ \text{VAL} & \begin{bmatrix} \text{SPR} & + \end{bmatrix} \end{bmatrix}$$

Given the rules and categories just sketched, it is important to see that our grammar now licenses trees like the one shown in (45):

(45)



Exercise 2: Understanding Tree (45)

A. For each node in (45) other than the preterminal nodes, identify the rule that licensed it.

B. Find the right abbreviation (e.g. NP, S, ...) for each node in (45).

Two rules we haven't yet reconsidered are the ones that introduce PP modifiers, repeated in (46):

(46) a. VP → VP PP
b. NOM → NOM PP

Although we will have nothing to say about the internal structure of PPs in this chapter, we would like to point out the potential for underspecification to simplify these rules, as well. Once categories are modeled as feature structures, we can replace the two CFG rules in (46) with one grammar rule, which will look something like (47):

(47) $\begin{bmatrix} \textit{phrase} \\ \text{HEAD} \quad \boxed{2} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textit{phrase} \\ \text{HEAD} \quad \boxed{2} \\ \text{VAL} \quad \begin{bmatrix} \text{SPR} & - \end{bmatrix} \end{bmatrix}$ PP

Note that the head daughter of this rule is unspecified for COMPS. In fact, all of the categories of type *phrase* licensed by our grammar are [COMPS itr], so specifying a COMPS value on the head daughter in addition to giving its type as *phrase* would be redundant.

Exercise 3: COMPS Value of Phrases

Look at the grammar summary in Section 3.6 and verify that this last claim is true.

In the next chapter, we will carry the collapsing of phrase structure rules even further. First, however, let us examine how features can be used in the analysis of agreement.

3.3.6 Representing Agreement with Features

In Section 3.3.3 above, we stated that the types *noun*, *verb* and *det* bear a feature AGR. In this section, we will consider what the value of that feature should be and how it can help us model subject-verb agreement.[15]

Agreement in English involves more than one kind of information. For subject-verb agreement, both the person and the number of the subject are relevant. Therefore, we want the value of AGR to be a feature structure that includes (at least) these two kinds of information, i.e. bears at least the features PER(SON) and NUM(BER). We will call the type of feature structure that has these features an *agr-cat* (*agreement-category*). The type *agr-cat* is a subtype of *feature-structure*.[16] The values of PER and NUM are atomic. The values of PER are drawn from the set {1st, 2nd, 3rd} and the values for NUM from the set {sg, pl}. The result is that instances of the type *agr-cat* will look like (48):

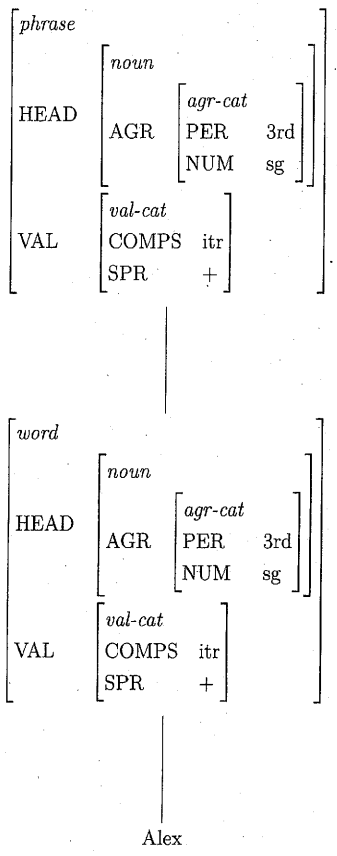
(48) $\begin{bmatrix} \textit{agr-cat} \\ \text{PER} & \text{3rd} \\ \text{NUM} & \text{sg} \end{bmatrix}$

[15] Determiner-noun agreement will be addressed in Problem 3 and then brought up again in Chapter 4.

[16] See the grammar summary in Section 3.6 for how this addition affects the type hierarchy.

AGR is a feature of (certain) subtypes of *pos*. This means that it is a HEAD FEATURE, i.e. one of the features that appears inside the HEAD value. Consequently, AGR-specifications get passed up from words to phrases and then to larger phrases. For example, the mother node in (49) will have the same specification for AGR as its head daughter:

(49)

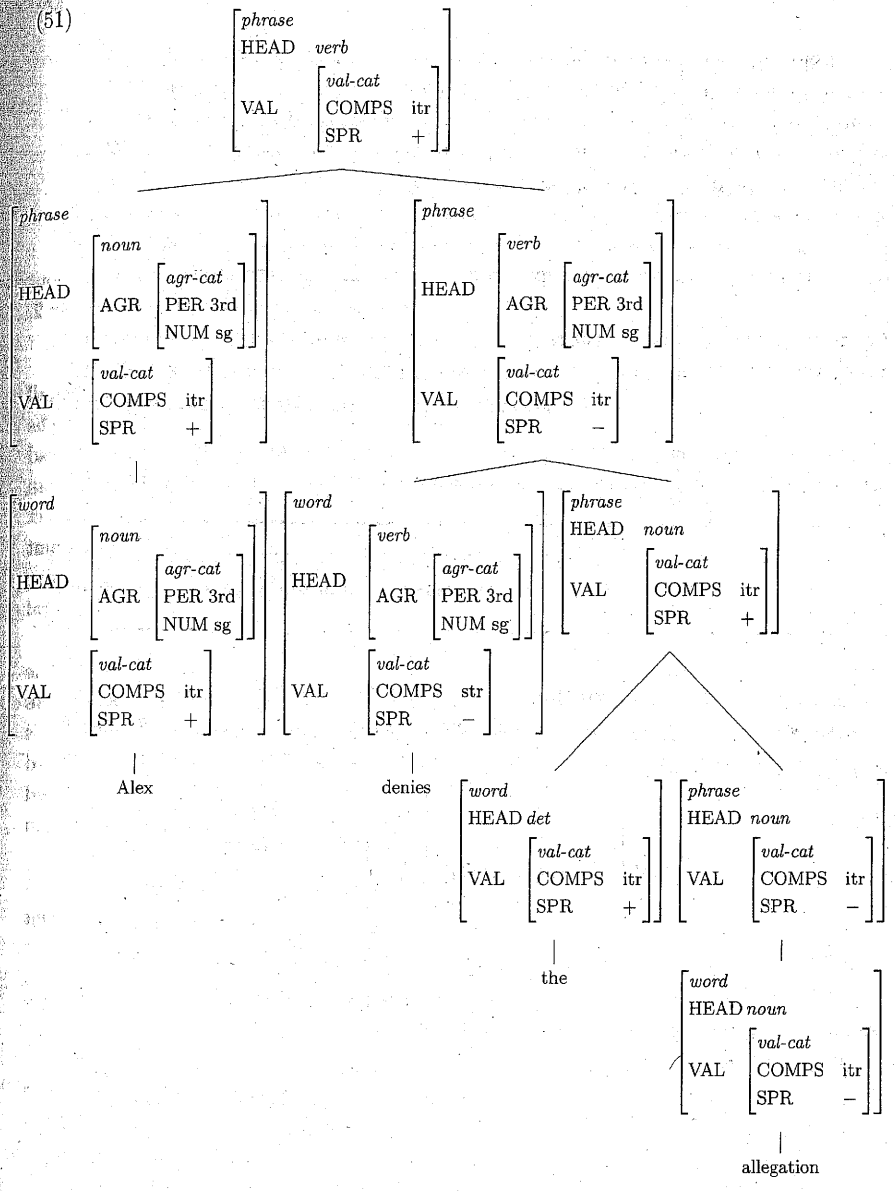


We want AGR information to be part of a phrase like this, because it is the kind of phrase that can be the subject of a simple sentence. If the verb within the VP and the noun that is the head of the subject NP both pass up their AGR specifications in this way, it is a simple matter to account for subject-verb agreement by revising our rule (40a) for combining NP and VP into an S. This revision may take the following form:

(50) $\begin{bmatrix} \textit{phrase} \\ \text{HEAD} \quad \boxed{1}\textit{verb} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \rightarrow \underset{\text{NP}}{\begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{AGR} & \boxed{2} \end{bmatrix} \end{bmatrix}} \begin{bmatrix} \textit{phrase} \\ \text{HEAD} \quad \boxed{1}\begin{bmatrix} \text{AGR} & \boxed{2} \end{bmatrix} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix}$

And in consequence of the revision in (50), AGR values are constrained as illustrated in (51):[17]

(51)



[17] In this tree, we omit the AGR specifications on the object NP and the root node, even though the grammar will provide them.

More generally, assuming the appropriate lexical entries, the revised analysis correctly accounts for all the contrasts in (52):

(52) a. The defendant denies the allegation.
b. *The defendant deny the allegation.
c. The defendants deny the allegation.
d. *The defendants denies the allegation.
e. The defendant walks.
f. *The defendant walk.
g. The defendants walk.
h. *The defendants walks.

Representing categories as complexes of features enables us to capture these facts without proliferating grammar rules. This is a distinct improvement over the CFG of Chapter 2.

3.3.7 The Head Feature Principle

The grammar rules proposed in the previous sections ((37a–c), (40), and (47)) have all identified the mother's HEAD value with the HEAD value of one of the daughters. The relevant HEAD-sharing daughter is always the one we have been referring to as the head daughter: the N in a NOM phrase, the NOM in an NP, the V in a VP, the VP in an S, and the VP or NOM that co-occurs with a PP modifier. But our theory does not yet include any notion of head daughter. If it did, we could factor out a general constraint about identity of HEAD values, instead of stating the same constraint in each of our five rules (with possibly more to come). The purpose of this section is to propose a general principle with this effect.

Rather than stipulating identity of features in an ad hoc manner on both sides of the rules, our analysis will recognize that in a certain kind of phrase – a HEADED PHRASE – one daughter is assigned special status as the HEAD DAUGHTER. Once this notion is incorporated into our theory (thus providing a remedy for the second defect of standard CFGs noted in the last chapter), we can factor out the identity constraint that we need for all the headed phrases, making it a general principle. We will call this generalization the Head Feature Principle (HFP).

Certain rules introduce an element that functions as the head of the phrase characterized by the rule. We will call such rules HEADED RULES. To indicate which element introduced in a headed rule is the head daughter, we will label one element on the right hand side of the rule with the letter '**H**'. So a headed rule will have the following general form:[18]

(53) [*phrase*] → ... **H**[ ] ...

So far, we have done two things: (i) we have identified the head daughter in a headed rule and (ii) we have bundled together (within the HEAD value) all the feature specifications that the head daughter must share with its mother. With these two adjustments in place, we are now in a position to simplify the grammar of headed phrases.

[18]Note that '**H**', unlike the other shorthand symbols we use occasionally (e.g. 'V' and 'NP'), does not abbreviate a feature structure in a grammar rule. Rather, it merely indicates which feature structure in the rule corresponds to the phrase's head daughter.

First we simplify all the headed rules: they no longer mention anything about identity of HEAD values:

(54) a. $\begin{bmatrix} \textit{phrase} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \mathbf{H} \begin{bmatrix} \textit{word} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix}$

b. $\begin{bmatrix} \textit{phrase} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \mathbf{H} \begin{bmatrix} \textit{word} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \text{str} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix}$ NP

c. $\begin{bmatrix} \textit{phrase} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \mathbf{H} \begin{bmatrix} \textit{word} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \text{dtr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix}$ NP NP

d. $\begin{bmatrix} \textit{phrase} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} & & \text{NP} \\ \text{HEAD} & \text{AGR} & \boxed{2} \end{bmatrix} \mathbf{H} \begin{bmatrix} \textit{phrase} \\ \text{HEAD} & \begin{bmatrix} \textit{verb} \\ \text{AGR} & \boxed{2} \end{bmatrix} \\ \text{VAL} & \begin{bmatrix} \text{SPR} & - \end{bmatrix} \end{bmatrix}$

e. $\begin{bmatrix} \textit{phrase} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \rightarrow \text{D} \ \mathbf{H} \begin{bmatrix} \textit{phrase} \\ \text{HEAD} & \textit{noun} \\ \text{VAL} & \begin{bmatrix} \text{SPR} & - \end{bmatrix} \end{bmatrix}$

f. $\begin{bmatrix} \textit{phrase} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \rightarrow \mathbf{H} \begin{bmatrix} \textit{word} \\ \text{HEAD} & \textit{noun} \\ \text{VAL} & \begin{bmatrix} \text{SPR} & + \end{bmatrix} \end{bmatrix}$

g. $\begin{bmatrix} \textit{phrase} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \mathbf{H} \begin{bmatrix} \textit{phrase} \\ \text{VAL} & \begin{bmatrix} \text{SPR} & - \end{bmatrix} \end{bmatrix}$ PP

The element labeled '**H**' in the above rules is the head daughter.

Second, we state the Head Feature Principle as a general constraint governing all trees built by headed rules.

(55) Head Feature Principle (HFP)
In any headed phrase, the HEAD value of the mother and the HEAD value of the head daughter must be identical.

The HFP makes our rules simpler by factoring out those properties common to all headed phrases, and making them conditions that will quite generally be part of the trees defined by our grammar. By formulating the HFP in terms of HEAD value identity, we allow information specified by the rule, information present on the daughter or the mother, or

information required by some other constraint all to be amalgamated, as long as that information is compatible.[19]

3.4 Phrase Structure Trees

At this point, we must address the general question of how rules, lexical entries and principles like the HFP interact to define linguistic structures. Our earlier discussion of this question in Chapter 2 requires some revision, now that we have introduced feature structures and types. In the case of simple context-free grammars, descriptions and structures are in simple correspondence: in CFG, each local subtree (that is, a mother node with its daughters) corresponds in a straightforward fashion to a rule of the grammar. All of the information in that local subtree comes directly from the rule. There is no reason to draw a distinction between the linguistic objects and the grammar's descriptions of them. But now that rules, lexical entries and principles like the HFP all contribute constraints (of varying degrees of specificity) that linguistic tokens must satisfy, we must take care to specify how these constraints are amalgamated and how the grammar specifies which expressions are grammatical.

3.4.1 The Formal System: an Informal Account

The distinction between descriptions and the structures they describe is fundamental. We use feature structures in our models of linguistic entities. Consider what this meant for the feature structures we used to model universities, departments and individuals. Each feature structure model was assumed to have all the properties relevant to understanding the university system; in our example, this included (for individuals) a name, a birthday, and a telephone number. The objects we took as models were thus complete in relevant respects.[20] Contrast this with descriptions of university individuals. These come in varying degrees of completeness. A description may be partial in not specifying values for every feature, in specifying only part of the (complex) value of a feature, in failing to specify a type, or in specifying nothing at all. A complete description of some entity will presumably be satisfied by only one thing – the entity in question. An empty description is satisfied by all the entities in the modeling domain. Any nonempty partial description is satisfied by some things in the modeling domain, and not by others.

Our theory of language works the same way. We use trees to model phrases and we use feature structures to model the grammatical categories that label the nodes in those trees. These models are complete (or RESOLVED) with respect to all linguistically relevant properties.[21] On the other hand, the lexical entries, grammar rules and principles are not models but rather partial descriptions of models. They thus need not be (and in

[19] The Head Feature Principle is sometimes formulated as 'percolation' of properties of lexical heads to the phrases that they 'project'. While it is often helpful to think of information as propagating up or down through a tree, this is just a metaphor. Our formulation of the generalization avoids attributing directionality of causation in the sharing of properties between phrases and their heads.

[20] Of course, a model and the thing it is a model of differ with respect to certain irrelevant properties. Our models of university individuals should omit any irrelevant properties that all such individuals presumably have, ranging from hair color to grandmothers' middle names to disposition with respect to Indian food.

[21] 'Resolvedness' is a direct consequence of our decision to define complex feature structures as total functions over a given domain of features.

fact usually aren't) fully resolved. For example, since the English word *you* is ambiguous between singular and plural, we might want to posit a lexical entry for it like the following:

(56)
$$\left\langle \text{you} , \begin{bmatrix} \textit{word} \\ \text{HEAD} & \begin{bmatrix} \textit{noun} \\ \text{AGR} & [\text{PER } 2\text{nd}] \end{bmatrix} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \right\rangle$$

This lexical entry is not complete in that it does not provide a specification for the feature NUM.[22]

Because the lexical entry is underspecified, it licenses two distinct WORD STRUCTURES (local, non-branching subtrees whose mother is of type *word*). These are shown in (57) and (58):

(57)
$$\begin{bmatrix} \textit{word} \\ \text{HEAD} & \begin{bmatrix} \textit{noun} \\ \text{AGR} & \begin{bmatrix} \textit{agr-cat} \\ \text{PER} & 2\text{nd} \\ \text{NUM} & \text{sg} \end{bmatrix} \end{bmatrix} \\ \text{VAL} & \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix}$$
|
you

(58)
$$\begin{bmatrix} \textit{word} \\ \text{HEAD} & \begin{bmatrix} \textit{noun} \\ \text{AGR} & \begin{bmatrix} \textit{agr-cat} \\ \text{PER} & 2\text{nd} \\ \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \\ \text{VAL} & \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix}$$
|
you

Here all the appropriate features are present (the mothers' feature structures are 'totally well-typed') and each feature has a completely resolved value.[23]

[22] This analysis of the ambiguity of *you* won't work in later versions of our grammar, and is presented here by way of illustration only.

[23] Again, this follows from defining feature structures in terms of total functions.

The relationship of the models to the grammar becomes more intricate when we consider not only lexical entries, but also grammar rules and the one general principle we have so far. These can all be thought of as constraints. Together, they serve to delimit the class of tree structures licensed by the grammar. For example, the grammar rule in (54b) above, repeated here as (59), is a constraint that can be satisfied by a large number of local subtrees. One such subtree is given in (60):

(59)
$$\begin{bmatrix} \textit{phrase} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \mathbf{H} \begin{bmatrix} \textit{word} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \text{str} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \quad \text{NP}$$

(60)
$$\begin{bmatrix} \textit{phrase} \\ \text{HEAD} \begin{bmatrix} \textit{verb} \\ \text{AGR} \begin{bmatrix} \textit{agr-cat} \\ \text{PER} & \text{2nd} \\ \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \\ \text{VAL} \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \textit{word} \\ \text{HEAD} \begin{bmatrix} \textit{verb} \\ \text{AGR} \begin{bmatrix} \textit{agr-cat} \\ \text{PER} & \text{2nd} \\ \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \\ \text{VAL} \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} & \text{str} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} \textit{phrase} \\ \text{HEAD} \begin{bmatrix} \textit{noun} \\ \text{AGR} \begin{bmatrix} \textit{agr-cat} \\ \text{PER} & \text{3rd} \\ \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \\ \text{VAL} \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix}$$

How many local subtrees are there that satisfy rule (59)? The answer to this question breaks down into a number of subquestions:

(61) a. How many feature structure categories can label the mother node?
b. How many feature structures categories can label the first daughter?
c. How many feature structures categories can label the second daughter?

The number of models satisfying (59) will be obtained by multiplying the answer to (61a) times the answer to (61b) times the answer to (61c), because, in the absence of other constraints, these choices are independent of one another.

Let us consider the mother node first. Here the types of the mother's and head daughter's feature structures are fixed by the rule, as are the SPR and COMPS values,

but the HEAD value is left unconstrained. In the grammar developed in this chapter, we have six parts of speech. This means that there are six options for the type of the HEAD value. If we pick *noun*, *det*, or *verb*, however, we have more options, depending on the values of AGR. Given that PER has three distinct values and NUM has two, there are six possible AGR values. Hence there are six distinct HEAD values of type *noun*, six of type *det* and six of type *verb*. Given that there is only one HEAD value of type *adj*, one of type *prep* and one of type *conj*, it follows that there are exactly 21 (= (3 × 6) + 3) possible HEAD values for the mother. Since all other feature values are fixed by the rule, there are then 21 possible feature structures that could label the mother node.

By similar reasoning, there are exactly 21 possible feature structures that could label the first (head) daughter in a local subtree satisfying rule (59). As for the second daughter, which is constrained to be an NP, there are only 6 possibilities – those determined by varying AGR values. Thus, there are 2646 (21 × 21 × 6) local subtrees satisfying rule (59), given the grammar developed in this chapter.

Note that one of these is the local subtree shown in (62), where the mother and the head daughter have divergent HEAD values:

(62) A Tree Not Licensed by the Grammar

$$\begin{bmatrix} \textit{phrase} \\ \text{HEAD} \begin{bmatrix} \textit{noun} \\ \text{AGR} \begin{bmatrix} \textit{agr-cat} \\ \text{PER} & \text{3rd} \\ \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \\ \text{VAL} \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \textit{word} \\ \text{HEAD} \begin{bmatrix} \textit{verb} \\ \text{AGR} \begin{bmatrix} \textit{agr-cat} \\ \text{PER} & \text{2nd} \\ \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \\ \text{VAL} \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} & \text{str} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} \textit{phrase} \\ \text{HEAD} \begin{bmatrix} \textit{noun} \\ \text{AGR} \begin{bmatrix} \textit{agr-cat} \\ \text{PER} & \text{3rd} \\ \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \\ \text{VAL} \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix}$$

It is subtrees like this that are ruled out by the HFP, because the HFP requires that the HEAD value of the mother be identical to that of the head daughter. Hence, by incorporating the HFP into our theory, we vastly reduce the number of well-formed local

subtrees licensed by any headed rule. The number of local subtrees satisfying both (59) and the HFP is just 126 (21 × 6). And in fact only 42 ((6 + 1) × 6) of these will ever be used in trees for complete sentences licensed by our grammar: in such trees, a word structure must be compatible with the head daughter, but only word structures for verbs or prepositions are ever specified as [COMPS str].

We complete the picture in much the same way as we did for CFGs. A phrase structure tree Φ is licensed by a grammar G if and only if:

- Φ is terminated (i.e. the nodes at the bottom of the tree are all labeled by lexical forms),
- the mother of Φ is labeled by S,[24]
- each local subtree within Φ is licensed by a grammar rule of G or a lexical entry of G, and
- each local subtree within Φ obeys all relevant principles of G.

A grammar is successful to the extent that it can be shown that the tree structures it licenses – its models – have properties that correspond to our observations about how the language really is. Recall from our discussion in Section 2.9 of Chapter 2 that what we are taking to be the reality of language involves aspects of both the mental representations of individual speakers and the social interactions among speakers. Thus, we're idealizing a fair bit when we talk about the sentences of the language being 'out there' in the world. In particular, we're abstracting away from variation across utterances and systematic variation across speakers. But we will have plenty to talk about before this idealization gets in our way, and we will have many observations and intuitions to draw from in evaluating the claims our models make about the external reality of language.

3.4.2 An Example

Consider the sentence *They swim.* Let's suppose that the lexical entries for *they* and *swim* are as shown in (63). Note that lexical entry for the plural form *swim* is underspecified for person.

(63) a.
$$\left\langle \text{they} , \begin{bmatrix} \textit{word} \\ \text{HEAD} & \begin{bmatrix} \textit{noun} \\ \text{AGR} & \begin{bmatrix} \text{PER} & \text{3rd} \\ \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \right\rangle$$

[24] Remember that S is now an abbreviation defined in (33) above.

b.
$$\left\langle \text{swim} , \begin{bmatrix} \textit{word} \\ \text{HEAD} & \begin{bmatrix} \textit{verb} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \right\rangle$$

Given these two lexical entries, the following are both well-formed local subtrees, according to our theory:

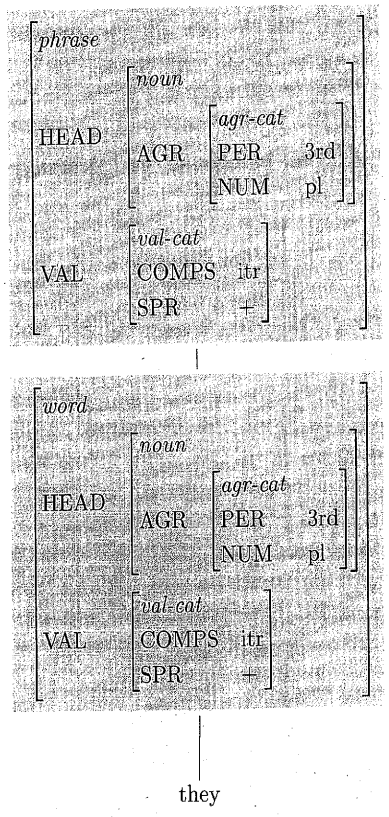
(64) a.
$$\begin{bmatrix} \textit{word} \\ \text{HEAD} & \begin{bmatrix} \textit{noun} \\ \text{AGR} & \begin{bmatrix} \textit{agr-cat} \\ \text{PER} & \text{3rd} \\ \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \\ \text{VAL} & \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix}$$
they

b.
$$\begin{bmatrix} \textit{word} \\ \text{HEAD} & \begin{bmatrix} \textit{verb} \\ \text{AGR} & \begin{bmatrix} \textit{agr-cat} \\ \text{PER} & \text{3rd} \\ \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \\ \text{VAL} & \begin{bmatrix} \textit{val-cat} \\ \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix}$$
swim

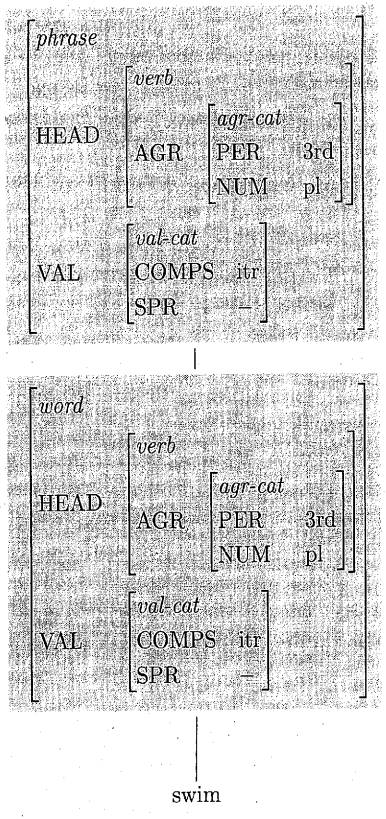
Observe that these word structures contain only fully resolved feature structures. Furthermore, the structure in (64b) contains a specification for the feature PER that will make the relevant tree structure compatible with the structure over *they* when we combine them to build a sentence.

These lexical structures can now be embedded within larger structures sanctioned by the rules in (54f,a) and the HFP, as illustrated in (65a,b):

(65) a.



b.



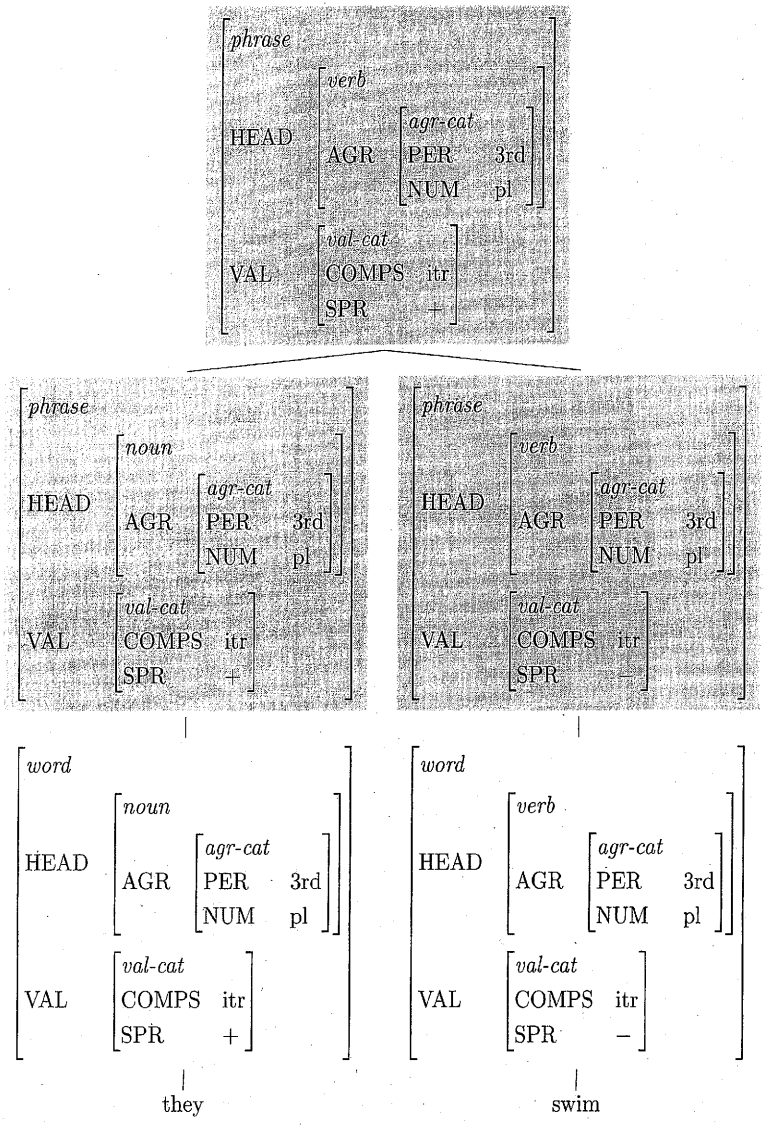
The shading in these and subsequent trees indicates the portion of the tree that is licensed by the rule in question (together with the HFP).

And finally, we can use rule (54d), repeated here as (66) to build a sentential phrase structure that combines the two previous structures. This is shown in (67):

(66)

$$\begin{bmatrix} \textit{phrase} \\ \text{VAL} \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \text{HEAD} \begin{bmatrix} \text{AGR} & \boxed{2} \end{bmatrix} \end{bmatrix}^{\text{NP}} \; \mathbf{H} \begin{bmatrix} \textit{phrase} \\ \text{HEAD} \begin{bmatrix} \textit{verb} \\ \text{AGR} \quad \boxed{2} \end{bmatrix} \\ \text{VAL} \begin{bmatrix} \text{SPR} & - \end{bmatrix} \end{bmatrix}$$

(67)

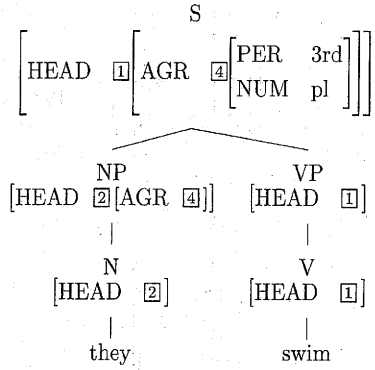


The nodes of the local subtree licensed by the rule in (66) (and the HFP) are again indicated by shading.

We will display phrase structure trees throughout this book, usually to illustrate the effect of particular constraints that are under discussion. Though the feature structures in the trees licensed by our grammar are always total functions, we will often display tree diagrams that contain defined abbreviations (e.g. NP or S) or which omit irrelevant feature specifications (or both). Similarly, we may want to illustrate particular identities

within phrase structure trees that have been enforced by linguistic constraints. To this end, we will sometimes include tags (e.g. 3) in our tree digrams to indicate identities induced by linguistic constraints. To illustrate the effect of the HFP, for example, we might replace the tree diagram in (67) with one like (68):

(68)



A diagram like (68) always abbreviates a phrase structure tree whose nodes are labeled by fully determinate, resolved feature structures.

3.5 Summary

The introduction of features has given us a formal mechanism for talking about ways in which sets of words (and phrases) behave both alike and differently. By allowing embedded feature structures, underspecifying categories, and formulating general constraints stating identities that must hold in well-formed trees, we have been able to generalize our phrase structure rules and reduce their number. This in turn has led us to carefully distinguish between our grammar rules and the fully determinate ('resolved') structures that satisfy them, and further between the models licensed by our grammar and the abbreviated representations of those models such as (68) that we will often use to focus our discussions throughout the remainder of this book.

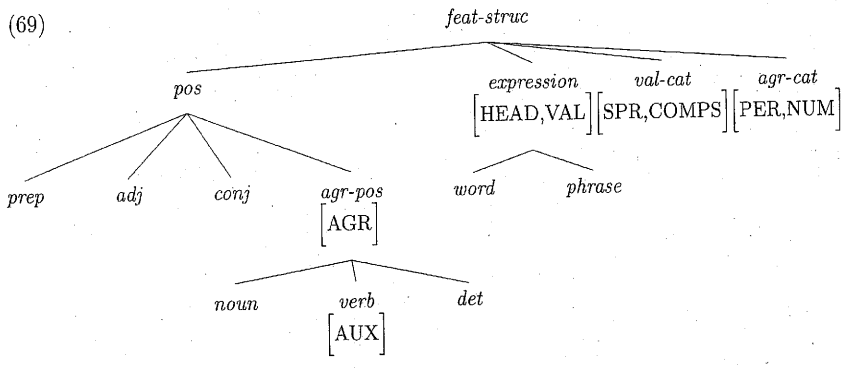
The theory we are developing is still closely related to standard CFG, yet it is somewhat more abstract. We no longer think of our phrase structure rules as specifying all the information that labels the nodes of trees. Rather, the rules, the lexicon, and some general principles – of which the HFP is the first example – all place certain constraints on trees, and any imaginable tree is well-formed so long as it conforms to these constraints. In this way, our grammar continues to be constraint-based, with the rules, lexical entries, and general principles all working together to define the well-formed structures of the language.

But the changes introduced in this chapter are not yet sufficient. They still leave us with three rules introducing complements that have too much in common and should be collapsed, and two rules introducing specifiers that similarly need to be collapsed. Moreover, as we will see in the next chapter, we have simplified the facts of agreement too much. The grammar we develop there will allow the more complex facts to be systematized, while at the same time eliminating further redundancy from the phrase structure rules of our grammar.

3.6 The Chapter 3 Grammar

3.6.1 The Type Hierarchy

(69)



3.6.2 Feature Declarations and Type Constraints

TYPE	FEATURES/CONSTRAINTS	IST
feat-struc		
pos		*feat-struc*
agr-pos	[AGR *agr-cat*]	*pos*
noun		*agr-pos*
det		*agr-pos*
verb	[AUX {+, −}]	*agr-pos*
prep		*pos*
adj		*pos*
conj		*pos*
expression	[HEAD *pos*; VAL *val-cat*]	*feat-struc*
word		*expression*
phrase		*expression*
val-cat	[COMPS {itr, str, dtr}; SPR {+, −}]	*feat-struc*
agr-cat	[PER {1st, 2nd, 3rd}; NUM {sg, pl}]	*feat-struc*

3.6.3 Abbreviations

(70)

$$\text{S} = \begin{bmatrix} \textit{phrase} \\ \text{HEAD} & \textit{verb} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \qquad \text{NP} = \begin{bmatrix} \textit{phrase} \\ \text{HEAD} & \textit{noun} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix}$$

$$\text{VP} = \begin{bmatrix} \textit{phrase} \\ \text{HEAD} & \textit{verb} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \qquad \text{NOM} = \begin{bmatrix} \textit{phrase} \\ \text{HEAD} & \textit{noun} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix}$$

$$\text{V} = \begin{bmatrix} \textit{word} \\ \text{HEAD} & \textit{verb} \end{bmatrix} \qquad \text{N} = \begin{bmatrix} \textit{word} \\ \text{HEAD} & \textit{noun} \end{bmatrix}$$

$$\text{D} = \begin{bmatrix} \textit{word} \\ \text{HEAD} & \textit{det} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix}$$

3.6.4 The Grammar Rules

(71) Head-Complement Rule 1:

$$\begin{bmatrix} \textit{phrase} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \mathbf{H} \begin{bmatrix} \textit{word} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix}$$

(72) Head-Complement Rule 2:

$$\begin{bmatrix} \textit{phrase} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \mathbf{H} \begin{bmatrix} \textit{word} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{str} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \text{ NP}$$

(73) Head-Complement Rule 3:

$$\begin{bmatrix} \textit{phrase} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \mathbf{H} \begin{bmatrix} \textit{word} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{dtr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \text{ NP NP}$$

(74) Head-Specifier Rule 1:

$$\begin{bmatrix} \textit{phrase} \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \rightarrow \overset{\text{NP}}{\begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{AGR} & \boxed{1} \end{bmatrix} \end{bmatrix}} \mathbf{H} \begin{bmatrix} \textit{phrase} \\ \text{HEAD} & \begin{bmatrix} \textit{verb} \\ \text{AGR} & \boxed{1} \end{bmatrix} \\ \text{VAL} & \begin{bmatrix} \text{SPR} & - \end{bmatrix} \end{bmatrix}$$

(75) Head-Specifier Rule 2:[25]

$$\begin{bmatrix} \textit{phrase} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \rightarrow \text{D} \quad \mathbf{H}\begin{bmatrix} \textit{phrase} \\ \text{HEAD} \quad \textit{noun} \\ \text{VAL} \quad \begin{bmatrix} \text{SPR} & - \end{bmatrix} \end{bmatrix}$$

(76) Non-Branching NP Rule:

$$\begin{bmatrix} \textit{phrase} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \rightarrow \mathbf{H}\begin{bmatrix} \textit{word} \\ \text{HEAD} \quad \textit{noun} \\ \text{VAL} \quad \begin{bmatrix} \text{SPR} & + \end{bmatrix} \end{bmatrix}$$

(77) Head-Modifier Rule:

$$\begin{bmatrix} \textit{phrase} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \rightarrow \mathbf{H}\begin{bmatrix} \textit{phrase} \\ \text{VAL} \quad \begin{bmatrix} \text{SPR} & - \end{bmatrix} \end{bmatrix} \text{PP}$$

(78) Coordination Rule:

$$\boxed{1} \rightarrow \boxed{1}^{+} \begin{bmatrix} \textit{word} \\ \text{HEAD} \quad \textit{conj} \end{bmatrix} \boxed{1}$$

3.6.5 The Head Feature Principle (HFP)

In any headed phrase, the HEAD value of the mother and the HEAD value of the head daughter must be identical.

3.6.6 Sample Lexical Entries

(79)

$$\left\langle \text{walks} , \begin{bmatrix} \textit{word} \\ \text{HEAD} \quad \begin{bmatrix} \textit{verb} \\ \text{AGR} \quad \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PER} & \text{3rd} \end{bmatrix} \end{bmatrix} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \right\rangle$$

$$\left\langle \text{walk} , \begin{bmatrix} \textit{word} \\ \text{HEAD} \quad \begin{bmatrix} \textit{verb} \\ \text{AGR} \quad \begin{bmatrix} \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \right\rangle$$

[25] See Problem 3 for more on this rule.

$$\left\langle \text{denies} , \begin{bmatrix} \textit{word} \\ \text{HEAD} \quad \begin{bmatrix} \textit{verb} \\ \text{AGR} \quad \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PER} & \text{3rd} \end{bmatrix} \end{bmatrix} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} & \text{str} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \right\rangle$$

$$\left\langle \text{deny} , \begin{bmatrix} \textit{word} \\ \text{HEAD} \quad \begin{bmatrix} \textit{verb} \\ \text{AGR} \quad \begin{bmatrix} \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} & \text{str} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \right\rangle$$

$$\left\langle \text{defendant} , \begin{bmatrix} \textit{word} \\ \text{HEAD} \quad \begin{bmatrix} \textit{noun} \\ \text{AGR} \quad \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PER} & \text{3rd} \end{bmatrix} \end{bmatrix} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & - \end{bmatrix} \end{bmatrix} \right\rangle$$

$$\left\langle \text{the} , \begin{bmatrix} \textit{word} \\ \text{HEAD} \quad \textit{det} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \right\rangle$$

$$\left\langle \text{Chris} , \begin{bmatrix} \textit{word} \\ \text{HEAD} \quad \begin{bmatrix} \textit{noun} \\ \text{AGR} \quad \begin{bmatrix} \text{PER} & \text{3rd} \\ \text{NUM} & \text{sg} \end{bmatrix} \end{bmatrix} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \right\rangle$$

$$\left\langle \text{you} , \begin{bmatrix} \textit{word} \\ \text{HEAD} \quad \begin{bmatrix} \textit{noun} \\ \text{AGR} \quad \begin{bmatrix} \text{PER} & \text{2nd} \end{bmatrix} \end{bmatrix} \\ \text{VAL} \quad \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \right\rangle$$

$$\left\langle \text{and} , \begin{bmatrix} word \\ \text{HEAD} & conj \\ \text{VAL} & \begin{bmatrix} \text{COMPS} & \text{itr} \\ \text{SPR} & + \end{bmatrix} \end{bmatrix} \right\rangle$$

3.7 Further Reading

One of the earliest (but often ignored) demonstrations of the descriptive power of feature structures is Harman 1963. Chomsky (1965) provides one of the earliest explicit discussions of syntactic features in generative grammar. The modern tradition of using complex feature structures (that is, features with feature structures as their values) begins with Kay 1979, Bear 1981, Bresnan 1982b, and Gazdar 1981 (see also Kaplan 1975 and Gazdar et al. 1985). For an elementary discussion of the formal properties of unification and its use in grammatical description, see Shieber 1986. For differing and more detailed technical presentations of the logic of typed feature structures, see King 1989, Carpenter 1992, Richter 1999, 2000, and Penn 2000.

3.8 Problems

⚠ Problem 1: Applying the Chapter 3 Grammar

A. Formulate a lexical entry for the word *defendants*.

B. Draw a tree for the sentence *The defendants walk*. Show the values for all of the features on every node and use tags to indicate the effects of any identities that the grammar requires.

C. Explain how your lexical entry for *defendants* interacts with the Chapter 3 grammar to rule out **The defendants walks*. Your explanation should make reference to grammar rules, lexical entries and the HFP.

Problem 2: 1st Singular and 2nd Singular Forms of Verbs

The sample lexical entry for *walk* given in (79) is specified as [AGR [NUM pl]]. This accounts for (i)–(iii), but not (iv) and (v):

(i) They walk.
(ii) We walk.
(iii) You (pl) walk. (cf. You yourselves walk.)
(iv) You (sg) walk. (cf. You yourself walk.)
(v) I walk.

Formulate lexical entries for *walk* in (iv) and (v). Be sure that those lexical entries don't license (vi):

(vi)*Dana walk.

⚠ Problem 3: Determiner-Noun Agreement

The Chapter 3 grammar declares AGR to be a feature appropriate for the types *noun*, *verb*, and *det*, but so far we haven't discussed agreement involving determiners. Unlike the determiner *the*, most other English determiners do show agreement with the nouns they combine with:

(i) a bird/*a birds
(ii) this bird/*this birds
(iii) that bird/*that birds
(iv) these birds/*these bird
(v) those birds/*those bird
(vi) many birds/*many bird

A. Formulate lexical entries for *this* and *these*.

B. Modify Head-Specifier Rule 2 so that it enforces agreement between the noun and the determiner just like Head-Specifier Rule 1 enforces agreement between the NP and the VP.

C. Draw a tree for the NP *these birds*. Show the value for all features of every node and use tags to indicate the effects of any identities that the grammar (including your modified HSR2) the Head Feature Principle requires.

Problem 4: Coordination and Modification

The Chapter 3 Grammar includes a coordination rule that is very similar to the coordination rule from the context-free grammar in (23) in Chapter 2 (see page 32).[26] The only difference is notational: Now that we have a more general kind of notation – tags for representing identity, we can replace the 'X's in the Chapter 2 version of the rule with tags.

The Chapter 3 Grammar also includes a Head-Modifier Rule. This rule corresponds to the two rules that introduced PPs in the Chapter 2 CFG:

(i) NOM → NOM PP
(ii) VP → VP PP

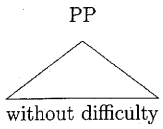
The first thing to notice about these rules is that they allow PPs to modify coordinate structures.[27] That is, the head daughter in the Head-Modifier Rule can be the entire italicized phrases in sentences like (iii) and (iv).

(iii) Alex *walks and reads books* without difficulty.
(iv) Terry likes the *poetry and music* on this program.

Of course, (iii) and (iv) are ambiguous: The PP can also be modifying just the rightmost conjunct within the coordinate structures.

[26] We will in fact revise this coordination rule in subsequent chapters.
[27] This was also true of the rules in the Chapter 2 grammar.

A. Draw the two trees for (iii) using the Chapter 3 grammar, and indicate which interpretation goes with which tree. [*Notes: You may use abbreviations for the feature structures at the nodes. Since we haven't given any sample lexical entries for prepositions, abbreviate the structure under the PP node with a triangle like this:*



The node above and may be abbreviated as CONJ.]

The Chapter 3 grammar, in its present form, doesn't allow PPs to modify Ss or NPs (which are both [SPR +]). Is this prediction correct? Consider the examples in (v) and (vi):

(v) Alex walks without difficulty.
(vi) Terry likes the music on the program.

In these examples, it is hard to tell which constituents the PPs *without difficulty* and *on the program* modify. Whether they attach low (modifying VP and NOM respectively, as currently permitted by the Chapter 3 grammar) or high (modifying S and NP, respectively, not currently permitted by the Chapter 3 grammar), we get the same string of words, and it's difficult to tell what the semantic differences between the two possible attachment sites would be. This question cannot be resolved just by considering simple examples like (v) and (vi).

B. Use coordination to resolve this question. That is, provide an argument USING EXAMPLES WITH COORDINATION to show that the prediction of the Chapter 3 grammar is incorrect: PPs must be able to modify S and NP as well as VP and NOM. [*Hint: Your argument should make reference to the different meanings associated with the different tree structures, depending on where the PP attaches.*]

Problem 5: Identifying the Head of a Phrase

The head of a phrase is the element inside the phrase whose properties determine the distribution of that phrase, i.e. the environments in which it can occur. We say that nouns head noun phrases, since (ii)-(v) can all show up in the same environments as (i): e.g. as the specifier of a verb, as a complement of a transitive verb and as the complement of prepositions like *of* or *on*.

(i) giraffes
(ii) tall giraffes
(iii) giraffes with long necks
(iv) all giraffes
(v) all tall giraffes with long necks

On the other hand (vi)–(ix) do not have the same distribution as the phrases in (i)–(v).

(vi) tall

(vii) with long necks
(viii) all
(ix) all tall

Thus it appears to be the noun in (i)–(v) that defines the distributional properties of the whole phrase, and it is the noun that we call the head.

In this problem we apply this criterion for identifying heads to a domain that is off the beaten path of grammatical analysis: English number names.[28] The goal of this problem is to identify the head in expressions like *two hundred* and *three hundred*. That is, which is the head of *two hundred*: *two* or *hundred*? In order to answer this, we are going to compare the distribution of *two hundred* with that of two minimally different phrases: *three hundred* and *two thousand*.

Now, many environments that allow *two hundred* also allow *three hundred* and *two thousand*:

(x) There were two hundred/three hundred/two thousand.
(xi) Two hundred/three hundred/two thousand penguins waddled by.

Some environments do distinguish between them, however. One such environment is the environment to the right of the word *thousand*:

(xii) four thousand two hundred
(xiii) four thousand three hundred
(xiv)*four thousand two thousand

A. Based on the data in (xii)–(xiv), which phrase has the same distribution as *two hundred*: *three hundred* or *two thousand*?

B. Does your answer to part (A) support treating *two* or *hundred* as the head of *two hundred*? Explain your answer in a sentence or two.

Similarly, we can compare the distribution of *two hundred five* to the two minimally different phrases *two hundred six* and *two thousand five*. Once again, the environment to the right of *thousand* will do:

(xv) four thousand two hundred five
(xvi) four thousand two hundred six
(xvii)*four thousand two thousand five

C. Based on the data in (xv)–(xvii), which phrase has the same distribution as *two hundred five*: *two hundred six* or *two thousand five*?

D. Does your answer to part (C) support treating *two hundred* or *five* as the head of *two hundred five*? Briefly explain why.

---

[28] This problem is based on the analysis of English number names in Smith 1999.