Nearness and Syntactic Influence Spheres

Marcus Kracht [1]
II. Mathematisches Institut
Freie Universität Berlin
Arnimallee 3
D – 14195 Berlin
`kracht@math.fu-berlin.de`

May 19, 1995

[1]This work has been carried out at the University of Potsdam within the *Innovationskolleg 'Kognitive Komplexität'* funded by the Deutsche Forschungsgemeinschaft.

Preface

This manuscript contains the first part of a larger manuscript of syntax, and replaces the *Theory of Syntactic Domains* and some versions of a mauscript called *Nearness and Syntactic Influence Spheres* that circulated during the period from 1992 to 1994. All these versions are now obsolete. Covering syntax in its entirety is an ambitious goal, and I appreciate any observations that my readers may have concerning the content. The present manuscript is worked out in detail only in the first three chapters. The remaining chapters are assembled from various own sources and therefore rather heterogeneous. I apologize for this. The reason is that I had to prepare the manuscript in time for the workshop, and this time was running out. I also apologize for not giving sufficient credits or references. I have meanwhile learnt that some of the results that I prove have been shown by other people, but a critical assessment would take too much time now.

I wish to thank Carsten Grefe and Paul Law for many discussions on this manuscript. Thanks also to Mark Johnson, Larry Moss, Martijn Spaan, Ed Stabler and Christian Wartena for helpful suggestions.

Berlin, May 19, 1995

Marcus Kracht

0

Contents

Chapter 1

Classification

In this chapter we will discuss one of the most fundamental techniques in linguistics, the *substitution test*. It is characteristic of the era of *structuralism* where it was declared to be the unique method for discovering the structure of a language. Although it has ceased to be unique in that respect, we will show that it is still quite a powerful tool especially when it comes to *correlations*, about which we will have much to say in the chapters to come.

1.1 A Short Outline of Structuralism

The most immediate concern of any scientific theory is that of *classification*. It goes hand in hand with abstraction, which is necessary for any kind of theory. Given the objects that we study we have to abstract away from part of the difference between various objects and concentrate on certain similarities instead. For example, we know that in English the letter 'p' stands for a certain sound, but each time we pronounce this sound, we will pronounce it differently. Thus, speaking of the sound 'p' is an abstraction, and this is the distinction that marks the difference between phonology and phonetics. For a phonetician, the different pronunciations of 'p' are real objects, but for a phonologist they constitute just the different realizations of a single abstract element, the *phoneme* $/p/$. A phonologist must of course assume that the phonemes are somehow identifiable, but does not particularly care how. Instead he goes on to describe in more detail the structure of the phonemes themselves and their relationship with each other. He observes, for example, that in addition to $/p/$ there is another phoneme $/b/$, and that the difference between the two is in a certain sense *minimal* and corresponds to the same difference that sets apart $/k/$ from $/g/$. Thus, he claims that the following following law of oppositions holds.

$$(1.1) \qquad \frac{/p/}{/b/} = \frac{/k/}{/g/}$$

By this it is expressed that the way $/p/$ corresponds to $/b/$ is the same way $/k/$ corresponds to $/g/$. We also speak of the 'quotient' $/p/ : /b/$ as an *opposition*. Unfortunately for a phonologist, he can neither say in phonological terms what it means that one element stands to another in the same way as a third to a fourth. But instead he must insist on the fact that this is something we can learn by asking around or asking ourselves (as a kind of informant; the judgements we obtain this way are said to be obtained by *introspection*, a common but somewhat dangerous tools in linguistics). All a phonologist can do after discovering this fact is to say that there is an abstract element, a feature which a phoneme can be either positively or negatively specified for, and if we assume that $/p/$ is positively specified for it, $/b/$ must be negatively specified for it, $/k/$ again positively and finally $/g/$ again negatively. By looking at the actual sounds we can also make a guess as to what this abstract feature corresponds to. It is the feature of being *voiceless*. We can identify this feature by the fact that it corresponds to the activity of the vocal chords in the production of this sound. If they are not active i. e. if they do not vibrate, then we have a [+*voiceless*] sound, otherwise a [−*voiceless*] sound. Returning to the equation above, we can say that the equation is true because whatever $/p/$ is, $/b/$ is the same with the only difference that the feature *voiceless* has a different value, and $/g/$ is the same as $/k/$ again except for the value of *voiceless*. If we now take another sound, say $/t/$, then we can put it above the line, so to speak, and ask what fills the fourth square of the equation.

$$(1.2) \qquad \frac{/p/}{/b/} = \frac{/t/}{/?/}$$

The answer in this case is $/d/$. Notice that it would be nonsense to put $/t/$ below, and so we learn that $/t/$ is specified for the feature in the same way as is $/p/$. Notice also that not for all sounds we can fill this equation. Although we find many other pairs, $/s/$ and $/z/$ and so on, there are sounds that resist to be put in either of the position. The most obvious example are vowels, but also $/h/$. It is not clear what we want to do in these cases. Since the elements are abstract from a phonologists point of view, there can be no real decision. However, we can use the phonetic criterion and then decide e. g. that vowels are always [−*voiceless*]. The reader should bear in mind that although for a phonologist the distinctive features are abstract because the analytic methods do not give an answer to the question of their (physical) constitution, no phonologist would deny the reality of the latter or refuse to allow it to himself as a guide for inquiry. But he would nevertheless insist, contrary perhaps to a phonetician, that the abstract elements are also real. So $/p/$ is not simply the set of all possible pronunciations or utterances of the sound $/p/$, but there is clearly a different sense in which we can say that $/p/$ exists. This is called for also because the set of such utterances is very large if not infinite and it would be foolish to assume that it is this set that we keep stored in our mind to identify $/p/$. Moreover, a phonologist would say that all that the listener cares for is the identification of the phonemes (or of words, or of message for that matter). He is not interested in the particular way $/p/$ gets pronounced. Moreover, the phonetic content of words is memorized by speakers not as executable motor programs for their pronunciation but rather as a an abstract sequence of phonemes, which individually get instantiated to motor programs if we wish to pronounce that very word. In this way, we need to store only once that $/p/$ is pronounced by putting the lips together, increasing the air pressure in the mouth and releasing it suddenly without activating the vocal chords. The word $/pet/$ contains no such information. It is simply stored as the sequence $/p/ + /e/ + /t/$ (see (Levelt, 1991)).

How does the linguist proceed from such equations establishing a distinctive feature? He says that $/b/$ differs from $/p/$ *ceteris paribus* with respect to the feature *voiceless*. He also says that they *differ minimally*, since there is only one feature whose assignment differs. Both, however, are true only in first approximation, and also only true in the English phoneme system. First, notice that French also contains a phoneme with the name $/p/$ and a phoneme with the name $/b/$. (Notice that a phonologist cannot tell whether French $/p/$ corresponds to English $/p/$. He has to use phonetic criteria here.) Using the same analysis in French it is established that there is roughly the same distinctive feature at work in both languages. Nevertheless, a difference exists. Namely, French voiceless stops are never aspirated, while in English (and German and other languages) they are. To represent the various distinctions between sounds across languages we can perform the same analysis; all we have to do is put all the phonemes of these languages together and ask for their difference. We will then have to represent English 'p' somewhat more accurately by $/p^h/$, while French 'p' is still denoted by $/p/$. So, we have the equation

(1.3) $$\frac{/p^h/}{/p/} = \frac{/t^h/}{/t/}$$

telling us that whatever distinguishes English 'p' from French 'p' is the same that distinguishes English 't' from French 't'. This is the abstract element *aspirated*. It is not a distinctive feature neither in English itself, nor in French. So if we now write somewhat more accurately for English

(1.4) $$\frac{/p^h/}{/b/} = \frac{/t^h/}{/d/}$$

then we can still say that the difference is one and only one feature, because although phonetically speaking $/p^h/$ differs from $/b/$ (at least) in the fact that it is voiceless and aspirated, we cannot have one difference without the other. It is – by the way – again a matter of choice that we say that *aspirated* is nondistinctive in English while *voiceless* is not. This can be stated only with recourse to the actual sounds. Namely, we can ask speakers to identify a nonaspirated $/p/$ and see whether they recongnize it as English 'p' or something else and likewise proceed with an aspirated voiced $/b^h/$ and look again whether it gets interpreted as a 'p' or as a 'b'. Now, finally, switch to Sanskrit. Here we have the following four sounds, $/p^h/$, $/p/$, $/b^h/$ and $/b/$. Moreover, the following equations hold

(1.5) $$\frac{/p^h/}{/p/} = \frac{/b^h/}{/b/} \qquad \frac{/p^h/}{/b^h/} = \frac{/p/}{/b/}$$

This establishes that both features are distinctive in Sanskrit, and that what is a minimal difference in English, the difference between $/p^h/$ and $/b/$, is a difference in two features rather than one in Sanskrit.

At the end of such an analysis we get a set of distinctive features and a mapping from combinations of these features (positively or negatively assigned) to phonemes. This map can be partial. There might be combinations that are illegitimate, for example, the combination of [+*voiceless*] with [+*vowel*]. These combinations are called *feature matrices* and are written in the following way

(1.6) $$\begin{bmatrix} + & \mathit{voiceless} \\ + & \mathit{labial} \\ - & \mathit{fricative} \\ + & \mathit{aspirated} \end{bmatrix}.$$

Such an analysis can be performed for specific languages, or for a group of languages. In the extreme case we can even develop a system for the sound inventory of all languages. The international phonetic alphabet (IPA) in a sense provides such a distinctive analysis of the totality of sounds in all human languages. Phonologists have also tried to see whether the feature hierarchy is structured, that is, whether some features control or preempt other features in the way that *voiced* is controlled by *vocal* and whether we can say something significant about this hierarchy. Furthermore, the fact that phonemes are put together in a string suggests we should look for the ways in which phonemes can or cannot be combined in a string. For we observe that even though all phonemes are pronounceable elements of a language, some linear combinations are impossible. For example, there are no voiced consonants at the end of a syllable in German or Russian (this phenomenon is called *devoicing*), and the onset of a syllable is mostly highly structured. For example, in English no syllable can begin with $/sr/$ or $/rp/$, but it can begin with $/spr/$ (as in *sprout*). Phonology is up to now the only discipline where the analysis in terms of distinctive features has been carried out to a satisfactory degree and a lot of knowledge about languages on the one hand and about this technique on the other has been accumulated. Syntax and semantics in comparison have too many primitive symbols to afford such an analysis, and it is up to now unclear what it can contribute in semantics. However, it can and is used successfully in syntax, and this is what interests us in the sections to come.

1.2 Feature Matrices as Boolean Vectors

Before carrying on with the informal development of Structuralism let us look in more detail into the formal presentation of phonemes. The idea we want to pursue here is to give a certain meaning to laws of opposition such as

(1.7) $$\frac{/p^h/}{/p/} = \frac{/b^h/}{/b/}.$$

We have explained informally what it means, but it is clear that beyond a mere formal reading of the operations of multiplication and division, allowing us as it were to derive the statement

$$\frac{/p^h/}{/b^h/} = \frac{/p/}{/b/} \tag{1.8}$$

there is no real sense in which we can say that we multiply or divide certain values. In fact there is also a different reading of these equations which we will propose here which in contrast to the present one is of an additive nature. It too is somewhat formal, but is also gives us an excuse to introduce certain important concepts from mathematics into the present discourse.

Recall that phonemes can be characterized as feature bundles, which means that we have a finite set of features and each phoneme is uniquely determined by giving each feature the value + or −. We can also think of the features as the axes in an abstract space, just as the x- and y-coordinate axes, so that a phoneme is just a vector in a multidimensional space with the values of the features being the values along the coordinate axes. However, these vectors have only two values to choose from for each dimension, so they are bit-vectors, to speak the language of computer science. In mathematical terms, they can be construed as vectors over a special field, the field $\mathbb{F}_2$ of integers modulo 2. This is a structure like the reals, consisting of two elements, 0 and 1, with addition + and multiplication · as follows.

(1.9)

+	0	1
0	0	1
1	1	0

·	0	1
0	0	0
1	0	1

It arises as follows. Take the integers and group them into two parts, E and O. E is the set of even numbers, and O the set of odd numbers. Write $E + E$ to denote the set of all sums $k + l$, where $k \in E$ and $l \in E$, and define $E + O$, $O + E$ and $O + O$ analogously. Then the following holds. $E + E = E$, i. e. the sum of two even numbers is even, $E + O = O + E = O$ and $O + O = E$. Likewise, $E \cdot O = E \cdot E = O \cdot E = E$ and $O \cdot O = O$. Now write 0 instead of E and 1 instead of O. What we get is exactly the tables shown above. Typically in mathematics one writes $k \equiv 0 \pmod 2$ to say that $k \in E$, and $k \equiv 1 \pmod 2$ to say that $k \in O$. The equation $k \equiv \ell \pmod 2$ generally abbreviates the fact that k leaves the same remainder as ℓ when divided by 2. The number 2 is called the *module*. In principle any number m can be used as a module, and we can in a similar way divide the set of integers into different sets, according to the remainder they leave when being divided by m. Call these sets R_0, $R_1, \dots, R_{m-1}$. Thus, with $m = 2$ we have $R_0 = E$ and $R_1 = O$. It can be shown that addition and multiplication acts classwise. But now let us return to $\mathbb{F}_2$. Addition has an inverse, that is, for every element k there is an element $-k$ such that $k + (-k) = 0$. In fact, we have $-k = k$, for $0 + 0 = 0$ as well as $1 + 1 = 0$. Thus, adding an element and substracting it yields the same result. (In fact, this is a special feature of the number 2.)

A vector space of dimension n over $\mathbb{F}_2$ is formed by taking the set of all n-long sequences of numbers from $\mathbb{F}_2$. There are 2^n such sequences. Vectors are denoted by arrows, such as in $\vec{v}$ and $\vec{w}$. Addition of these sequences is coordinatewise. For example, with $n = 3$ we have $\langle 0,1,1\rangle + \langle 1,1,1\rangle = \langle 1,0,0\rangle$. The vector $\vec{0}$, all of whose entries are 0, plays a special role in that $\vec{v} + \vec{0} = \vec{v}$. We say that it is the *neutral element* with respect to addition. As in $\mathbb{F}_2$, addition is the same as substraction, because $\vec{v} + \vec{v} = \vec{0}$. The vectors $\vec{e_1} = \langle 1,0,0,\ldots\rangle$, $\vec{e_2} = \langle 0,1,0,\ldots\rangle$, $\vec{e_3} = \langle 0,0,1,\ldots\rangle$ etc. have the property that each vector can be represented as the unique sum of a subset of these vectors. We say that they form a *basis*. This is not the only basis we can find for a vector space, in fact there are quite a lot. In three dimensions, for example, $\langle 1,1,0\rangle$, $\langle 0,1,1\rangle$ and $\langle 1,1,1\rangle$ form a basis.

Let us now return to our analysis of phonemes. To be concrete, fix a set of features, such as *voiced*, *aspirated*, *labial* and *dental*. This gives a four-dimensional space and in total 16 different phonemes. Not all of them might be realized in a language, in fact a phoneme which is [+*dental*] is most likely [−*labial*]. In Sanskrit, for example, we find all remaining twelve of the sixteen combinations realized. Let us make our life simple by restricting to a specific set of phonemes in Sanskrit, namely to the following set

(1.10)

	voiced	*aspirated*	*labial*	*dental*
$/p/$	−	−	+	−
$/p^h/$	−	+	+	−
$/b/$	+	−	+	−
$/b^h/$	+	+	+	−
$/t/$	−	−	−	+
$/t^h/$	−	+	−	+
$/d/$	+	−	−	+
$/d^h/$	+	+	−	+
$/k/$	−	−	−	−
$/k^h/$	−	+	−	−
$/g/$	+	−	−	−
$/g^h/$	+	+	−	−

Now, given the features and the assignment of features, we can define a map d from phonemes into the four-dimensional vector space $\mathbb{F}_2^4$. Namely, the first coordinate of $d(x)$ is 0 whenever the phoneme x is [−*voiced*], and it is 1 whenever x is [+*voiced*]; the second coordinate is 0 whenever x is [−*aspirated*] and 1 otherwise; and so on.

We can now give a rather precise interpretation for the equations of the previous section. Namely, instead of

$$\frac{/p^h/}{/p/} = \frac{/b^h/}{/b/} \tag{1.11}$$

we can now write

$$d(/p^h/) - d(/p/) = d(/b^h/) - d(/b/). \tag{1.12}$$

This means that whatever the difference is between $d(/p^h/)$ and $d(/p/)$, it is the same difference as between $d(/b^h/)$ and $d(/b/)$. Notice that the difference is $\vec{0}$ iff the two have identical feature assignments. Given the laws of arithmetic in $\mathbb{F}_2$, adding is the same as taking away, so the equation can be rewritten into

$$d(/p^h/) + d(/p/) = d(/b^h/) + d(/b/). \tag{1.13}$$

Furthermore, we can add the left hand side on both sides and obtain a fully symmetrical variant of the law of oppositions.

$$d(/p^h/) + d(/p/) + d(/b^h/) + d(/b/) = \vec{0}. \tag{1.14}$$

This is a more abstract version, and it is actually less informative because it permits us to deduce that the following laws are equivalent.

$$\frac{/p^h/}{/p/} = \frac{/b^h/}{/b/} \quad \Leftrightarrow \quad \frac{/p^h/}{/b^h/} = \frac{/p/}{/b/} \quad \Leftrightarrow \quad \frac{/p^h/}{/b/} = \frac{/b^h/}{/p/} \tag{1.15}$$

The first equivalence also follows if we adopt a purely formal attitude to this equation, and 'multiply' by $/p/$ and then divide $/b^h/$. Obviously, this is also the way in which the equation is meant to be understood. Notice that the 'additive' reading presented here, allows to deduce more equations because there is no directionality in the difference. The original equation says not only that the differences are equal, but also that the values assigned to the upper left element is the same as the assignment to the upper right element. So, in the multiplicative interpretation it can be deduced that the value of *aspirated* is equal for $/p^h/$ and $/b^h/$, while it is not deducible which value it actually has. In the additive reading it can only be deduced that we have two aspirated elements and two non-aspirated elements in the equation. Nevertheless, the structure of the assignment is fixed up to isomorphism by all the laws of opposition that we can write down. Not all of them are as meaningful. For example, we can write

$$d(/p^h/) + d(/p/) = d(/k^h/). \tag{1.16}$$

Furthermore, the phoneme $/k/$ acts as the neutral element in these equations. The reason for this apparent nonsense is that these equations are not invariant under switching the polarity of the features, which they must be if they are true laws of opposition. By switching the polarities of the features (that is, making $-$ what has been $+$ and conversely) we can turn any element into the neutral element of addition. To avoid it figuring in the laws we must require that laws are true laws of opposition, that is, they must contain an even number of terms. In phonology, where we know exactly what the difference is between phonemes the additive laws of opposition might seem just a formal device, but it is important to note that in absence of any knowledge for the 'direction' or polarity of an opposition (that is, $+$ above the line and $-$ below versus $-$ above the line versus $+$ below) all that we can say is that the feature constituting the difference in one opposing pair is the same as in the second.

An important concept to be introduced here is the so-called *Hamming-distance*, δ_H. It is calculated as follows. Two vectors $\vec{v}$ and $\vec{w}$ have the Hamming-distance k if they differ at exactly k places. For example, $\delta_H(\langle 0, 1, 1, 0\rangle, \langle 1, 0, 1, 1\rangle) = 3$. An opposition between two phonemes is called *privative* if they have the Hamming-distance 1. Notice that whether or not an opposition is privative depends on the coordinatization, that is, on the features that we use to analyze. For example, the opposition between $/p^h/$ and $/b/$ is privative in Sanskrit but not in English. If we were to introduce a different set of distinctive features we can arrange it that the opposition is non-privative also in Sanskrit. For example, let us introduce *strong* to mean either both voiceless and aspirated or voiced and nonaspirated. Let the strong feature replace the voiced feature. Then $/p^h/$ is strong and voiceless, $/b/$ is strong and voiced. So, they differ only with respect to a single feature; their Hamming-distance is now 1 and the opposition is privative.

1.3 Paradigmatic and Syntagmatic Relations

Let us switch from the sound inventory of a language to look at the inventory of words. We can carry out the same kind of analysis as with sounds, but syntax also allows a different approach, one which has not so intensively been explored. First of all, words can be put together to larger units, sentences or other. Let us for the moment restrict to sentences. The elements in a sentence enter a relationship by the fact that they are now in the same string of words. Thus, in

(1.17) *The cat is on the mat.*

it is the word /cat/ that finds itself to the right of /the/ and to the left of /is/ and so on. [1] On the other hand, the other words in the language can compete with the ones in the string for their respective position in that string. So, /hat/ competes with /cat/, and also with /mat/. /was/ competes with /is/, /this/ with /the/ and so on. We say that /the/ and /mat/ enter a *syntagmatic relationship* and that /hat/ and /mat/ enter a *paradigmatic relationship*. But what does that actually mean? Can we say that /the/ can enter a syntagmatic relationship with /was/ as in

(1.18) *The was in cat sat on the.*

Or can we really say that /hat/ competes with /cat/ in any sense? Well, the fundamental notion that connects these issues is that of grammatical acceptability. To say that elements can enter a syntagmatic relationship means that they can be part of a syntactically acceptable string in a given way. So, we see that /the/ cannot be directly to the left of /was/ because there is no grammatically acceptable sentence in which that is so. Likewise, /hat/ competes with /cat/ only insofar as the first can be exchanged for the latter without altering the syntactic acceptability of the sentence. It is hard to justify such things, of

[1]In analogy to phonology, we write /cat/ to denote the word which is usually written *cat*. However, by using the slashes we abstract away from irrelevant details, notably the spelling (at the beginning of a sentence it is written *Cat*), the use of different letter fonts, letter sizes and more of that size. Finally, it is also immaterial whether this word is spoken, written or identified in some other way.

course, because we are talking about an infinite set of sentences, and we have no obvious means to tell whether or not these two claims just made are true. Moreover, syntactic acceptability mostly consists in a grade of acceptability, that is, we judge sentences not simply as acceptable or unacceptable, but as more or less acceptable, as slightly odd etc. Throughout this book we will not be concerned with graded judgements, although we concede that this is a limitation of the whole analysis. Now, given that there is a way to classifiy sentences into the two categories, syntactically acceptable and syntactically unacceptable, we can now take the entire vocabulary of a language, consisting of all words in all possible fully inflected forms, and subject it to an analysis according substitutivity in all contexts. Let us make precise how this can be achieved.

In the same way that phonology has *phonemes* rather than sounds, the syntax has *lexemes* rather than words. The notion of a lexeme allows to abstract away from different appearances, such as due to dialectal variation or different spelling, see the footnote above. We let V to be the *vocabulary* of a language, now consisting of *lexemes*. In the standard literature on formal languages this set is referred to as the *alphabet*. This is just a matter of convenience, but it means that what is called a *word* there must be called a *string* here to avoid confusion. Strings over V are formed by linear juxtaposition or *concatenation*. We denote the concatenation of the string $\vec{v}$ and $\vec{w}$ sometimes by $\vec{v} \cdot \vec{w}$ but mostly by $\vec{v}\vec{w}$. The empty string is denoted by ϵ. V^* is the set of all strings that can be formed by concatenation of ϵ with elements of V, in one step, two steps, and so on, thus, in n steps for any natural number n. A *string* is an element of V^*. Strings are either symbolized by simple letters or by an arrow, such as in $\vec{w}$. A *context* is a pair $C = \langle \vec{w}_1, \vec{w}_1 \rangle$, where $\vec{w}_1$ and $\vec{w}_2$ are strings. In a string $\vec{s}$ a substring $\vec{x}$ occurs in the context $C = \langle \vec{w}_1, \vec{w}_2 \rangle$ if $\vec{s} = \vec{w}_1 \cdot \vec{x} \cdot \vec{w}_2$. Strings can have multiple occurrences of the same word or symbol. Now, a *language* over V is for our purposes at hand simply a subset $L \subseteq V^*$. A string $\vec{v}$ is *acceptable* in L iff $\vec{v} \in L$, and unacceptable otherwise. Modulo L we say that x *occurs* or *can occur* in the context $C = \langle \vec{w}_1, \vec{w}_2 \rangle$ if $\vec{w}_1 \cdot \vec{x} \cdot \vec{w}_2 \in L$. Now, we say that $\vec{x}$ and $\vec{y}$ are *syntactically equivalent* iff for all contexts C, $\vec{x}$ occurs in C iff $\vec{y}$ occurs in C. In symbols we write $\vec{x} \approx_L \vec{y}$. This is the formal equivalent of the notion of *competition*. We can then go on to classify the elements of V according to syntactical equivalence. However, this method can fail to give satisfactory results. For example, consider the words /fire/ and /hat/. We want to say that they are syntactically equivalent in English, even though it is not always meaningful to interchange them. That means that we focus on what we describe by *syntactical acceptability* regardless of whether a sentence makes sense or not. In the sentence

(1.19) *His boss wants to fire him because of his stupidity.*

we cannot replace /fire/ by /hat/ for this would yield a violation in syntactic acceptability, not only in (semantic) interpretability. Standardly, we assume that /fire/ is actually not a single word but two, say, /fire$_1$/ and /fire$_2$/ only one of which is fully substitutable with /hat/. With hindsight we can say that one is a verb and the other is a noun. But crucially, we want our test to give us this result. The clue to the solution is to use the notion of *syntactic content*. We say that $\vec{x}$ *has at least the syntactic content* of $\vec{y}$, in symbols $\vec{x} \preccurlyeq_L \vec{y}$, if in every context in which $\vec{x}$ occurs, $\vec{y}$ can also occur. Alternatively, we can associate with each word $\vec{x}$ the set $\mathfrak{C}_L(\vec{x})$ of all contexts in which $\vec{x}$ occurs. Then $\vec{x} \preccurlyeq_L \vec{y}$ iff $\mathfrak{C}_L(\vec{x}) \subseteq \mathfrak{C}_L(\vec{y})$. Moreover, $\vec{x} \approx_L \vec{y}$ iff $\mathfrak{C}_L(\vec{x}) = \mathfrak{C}_L(\vec{y})$ iff $\vec{x} \preccurlyeq_L \vec{y}$ and $\vec{y} \preccurlyeq_L \vec{x}$. Returning to our example, we have /hat/ $\preccurlyeq_E$ /fire/, but the two are not equivalent. Also, we have /see/ $\preccurlyeq_E$ /fire/ (taking here only into account that both are transitive verbs; whether they are truly syntactically identical or has to be left open, but is of no significance for the argument). Moreover, we observe that

$$\mathfrak{C}_E(\text{/fire/}) = \mathfrak{C}_E(\text{/see/}) \cup \mathfrak{C}_E(\text{/hat/}). \tag{1.20}$$

Additionally, we have

$$\mathfrak{C}_E(\text{/see/}) \cap \mathfrak{C}_E(\text{/hat/}) = \emptyset. \tag{1.21}$$

The last says that there are two classes of words which occur in strictly separate contexts. /fire/ extends to both classes, and this motivates positing two distinct words, /fire$_1$/ and /fire$_2$/. The identity with one of the two can only be established by the context, for we define them by the equation

$$\begin{array}{lcl} \mathfrak{C}_E(\text{/fire}_1\text{/}) & = & \mathfrak{C}_E(\text{/fire/}) \cap \mathfrak{C}_E(\text{/hat/}) \\ \mathfrak{C}_E(\text{/fire}_2\text{/}) & = & \mathfrak{C}_E(\text{/fire/}) \cap \mathfrak{C}_E(\text{/see/}) \end{array} \tag{1.22}$$

In other words, /fire/ is /fire$_1$/ exactly in those contexts, in which it can be substituted by /hat/ without altering the syntactic acceptability, and it is used as /fire$_2$/ in those contexts where it can be replaced by /see/ without changing the synactic acceptability.

What we have just described is the discovery procedure for word-classes. It is strictly speaking also the procedure which has to be applied when isolating *phonemes*. In the previous section we have taken for granted that it can be done, but in syntax this is not obviously so, again because the vocabulary of words is large and also flexible, while the sounds of a language counts well below a hundred, and are relatively stable. We will for the moment take for granted that the procedure as outlined above yields such classes in a clear way and we attach to them meaningful labels such as *noun*, *intransitive verb*, *complementizer* etc. As discussed, the context sets of lexemes need not be disjoint, and so the picture is not such a simple one. Nevertheless, given that we have only finitely many lexemes, there is a finite number of context set $\mathfrak{C}_E(x)$ generated from lexemes (but a possibly infinite one created by strings of lexemes). These are the basic context sets, corresponding to naive word classes. If we want to avoid the apparent problems as with the word /fire/, we need to consider all possible intersections of these basic contexts sets, and take all minimal such intersections. Let us call them *basic types*. A basic type is *inhabited* if it is the context of a lexeme, i. e. of the form $\mathfrak{C}_L(v)$ for some $v \in V$ rather than $\mathfrak{C}_L(\vec{x})$. It is not a priori clear that each basic type is inhabited. We can think of strings whose set of common contexts is not the context set of a simple lexeme. But there is a natural expectation is that this cannot arise. We will not push this issue further. If the reader wants an example, think of the various selection possibilities of a verb. A verb may select two noun phrase objects, or a noun phrase and a prepositional phrase of some kind. By saturating one or more arguments we obtain complex verb phrases which look for quite different argument sets. Thus is the case in *John gives a book ...* which requires a prepositional phrase headed by *to*. So it belongs to the context set of /to/ + /john/, but not to the context set of any single lexeme. Let us, however, now focus on the analysis in the form of distinctive pairs. For example, in Latin the following equations are valid

(1.23)
$$\frac{/\text{poetae}/}{/\text{poeta}/} = \frac{/\text{ranae}/}{/\text{rana}/}$$
$$\frac{/\text{poetae}/}{/\text{poeta}/} = \frac{/\text{sigilla}/}{/\text{sigillum}/}$$
$$\frac{/\text{poetae}/}{/\text{poeta}/} = \frac{/\text{rēs}/}{/\text{rēs}/}$$

However, again we must ask what precisely this means. There is no obvious way in which we can use the contexts here as a criterial device, so we are left with only our intuition to tell us that there is a common difference established by these equations. Being simple minded, we can point to the first equation and say that what is the common difference is the addition of an /e/. But this is refuted by the second, and more obviously even by the third, in which there is no overt difference. If we take the meaning into the account here we can say that the difference that is at issue is the difference between singular and plural. (The first equation can also be used to establish the difference dative/nominative, but let us ignore that here.) So we can as in the contrastive analysis of phonemes posit a feature to express the common difference.

Nevertheless, there is a syntactic way to get at these distinctions. Consider a pair of two strings, $\vec{x}_1$ and $\vec{x}_2$. A context for such a pair is a triple $C = \langle \vec{w}_1, \vec{w}_2, \vec{w}_3 \rangle$, and we say that $\langle \vec{x}_1, \vec{x}_2 \rangle$ occurs in the string $\vec{s}$ in the context C iff $\vec{s} = \vec{w}_1 \cdot \vec{x}_1 \cdot \vec{w}_2 \cdot \vec{x}_2 \cdot \vec{w}_3$. With respect to a language L we say that $\langle \vec{x}_1, \vec{x}_2 \rangle$ *can occur* in C if $\vec{w}_1 \cdot \vec{x}_1 \cdot \vec{w}_2 \cdot \vec{x}_2 \cdot \vec{w}_3 \in L$. Now consider another pair $\langle \vec{y}_1, \vec{y}_2 \rangle$. If it occurs in the context C just as $\langle \vec{x}_1, \vec{x}_2 \rangle$ then $\vec{y}_1$ competes with $\vec{x}_1$ and $\vec{y}_2$ competes with $\vec{x}_2$. For example, take ⟨/hat/, /sat/⟩ and ⟨/cat/, /slept/⟩. Then it is /hat/ which competes with /cat/ for its syntactic position, and /slept/ which competes with /sat/. Both are actually individually competitors. However, this competition can be incomplete, as in ⟨/cats/, /sit/⟩ and ⟨/cat/, /sits/⟩. Here, considering a context where we can replace /cat/ by /cats/ we cannot do this without also changing /sits/ to /sit/. We express this by saying that means that there is a correlation between the pair {/cats/, /cat/} and the pair {/sits/, /sit/}. A correlation can be established on the basis of a single context, such as

(1.24) *The ... ... on the mat.*

(Notice that in this context there is no element separating the first and the second element. That is, in this context $\vec{w}_2 = \epsilon$.) Different contexts may not establish this correlation, as

(1.25) *The ... would like to ... on the mat.*

To put it once again, we have two elements, $\vec{x}_1$ and $\vec{x}_2$, competing for the same position in a string, and two elements $\vec{y}_1$ and $\vec{y}_2$ competing for a (disjoint) position. We observe that of the four possible combinations in a given context, only two are legal:

(1.26) $$\begin{array}{llll} \vec{w}_1 \cdot \vec{x}_1 \cdot \vec{w}_2 \cdot \vec{x}_2 \cdot \vec{w}_3 & \surd & \vec{w}_1 \cdot \vec{x}_1 \cdot \vec{w}_2 \cdot \vec{y}_2 \cdot \vec{w}_2 & * \\ \vec{w}_1 \cdot \vec{y}_1 \cdot \vec{w}_2 \cdot \vec{x}_2 \cdot \vec{w}_3 & * & \vec{w}_1 \cdot \vec{y}_1 \cdot \vec{w}_2 \cdot \vec{y}_2 \cdot \vec{w}_3 & \surd \end{array} .$$

In that particular situation we will say that there is an abstract element which distinguishes $\vec{y}_1$ from $\vec{x}_1$, and that same feature distinguishes $\vec{y}_2$ from $\vec{x}_2$. This abstract element can be characterized via the meaning of these elements. It corresponds to the distinction between *singular* and *plural*. So, we have established that there are singular verbs, plural verbs, and that there are singular nouns and plural nouns. We can go on with this search and discover that the singular/plural distinction is at work in the contrast /this/ : /these/ and so on. We will furthermore postulate that in the context that established the correlation it is only elements with the same value for the feature *singular* that can cooccur. This accounts for the well-known phenomenon of subject-verb agreement and also agreement between an adjective and a modified noun. Further investigation will reveal that there also is what we know as *gender* and *case* in English. There are three genders, so that an analysis in terms of binary features is not straightforward. But more on that later. What we have shown is that in analysing the way in which elements combine with each other we can establish an internal structure of these elements that we would not be able to see if we were to analyse the elements on their own. To return to phonology to see the relevance of this. Notice that there we had phonetic descriptions that enabled us to see that there is a common regularity in the oppositions $/p/ : /b/$, $/t/ : /d/$ and $/k/ : /g/$ and that in turn allowed to establish the feature analysis of phonemes. But what good is that for if the only thing it show is that the elements are distinct? Well, look at the phenomenon of final devoicing in German. What we find is that we have the verb /reisen/, in which the sibilant – being syllable initial – is pronounced $/z/$. At the end of the syllable it is pronounced $/s/$. So, we have the 3^{rd} singular present /reist/, in which the sibilant is voiceless, and so is the $/t/$. Similarly, we have the noun /jagd/ derived from /jagen/. The ‘g’ is voiceless in the first, and voiced in the second. We observe that the end of the syllable may contain several consonants, but they all must be voiceless. Without the notion of a voiced/voiceless phoneme we would have to express that same observation with a cooccurrence list. Linguists would say that we are missing a generalization here, because it is quite obvious what common property the consonants have that can cooccur. Another case in this connection is that of *vowel harmony* which can be found in Finnish, Hungarian and Turkish, for example. In Finnish, within a word vowels must agree in their *frontness*. There are three sets of oppositions, and we have

(1.27) $$\frac{/\ddot{a}/}{/a/} = \frac{/\ddot{o}/}{/o/} = \frac{/y/}{/u/}.$$

In each case the upper vowel is pronounced with advanced tongue while the lower is not. So we have two series, $\{\ddot{a}, \ddot{o}, y\}$ and $\{a, o, u\}$. $/e/$ and $/i/$ can cooccur with either of the series. So the following words are well-formed

(1.28) *päällikkö*, *kansallis*, *osake*, *säännöttömys*, *kaupunki*

To express this law we have to make use of the feature *front*. We leave the problem of how to incorporate the behaviour of /*e*/ and /*i*/. There certainly are ways to do this, but the statement of the law is not quite simple even with the help of the feature *front*. The point we wanted to make in this section is nevertheless clear. It is that the distinctive features of the contrastive analysis can only be intrinsically shown to arise if we look at syntagmatic relationships, and this is also what these features can help to explain.

1.4 Sets, Features and Boolean Algebras

The formal object used for studying features is that of a *boolean algebra*. We will introduce boolean algebras via sets and prove later that the laws of boolean algebra allow to build for each abstract boolean algebra an isomorphic algebra using sets. The concrete boolean algebras are formed from an arbitrary set X. The subsets of X are collected in the so-called *powerset* $\wp(X)$. On $\wp(X)$ we have the operations of *intersection* $\cap$, *union* $\cup$ and *relative complement* $-$. The first two are defined for arbitrary sets, and it is easily seen that the intersection of two subsets of X is a subset of X, and so is the union. The relative complement $-U$ of a set $U \in \wp(X)$ is defined by $X - U$, a set containing all elements of X which are not in U. A *field of sets* is a sextuple $\langle D, \cap, \cup, -, \emptyset, X\rangle$ where D is a subset of $\wp(X)$ which is closed under the three operations, and contains $\emptyset$ as well as X. In particular, the powerset algebra $\mathbf{2}^X := \langle \wp(X), \cap, \cup, -, \emptyset, X\rangle$ is a field of sets. Any set of subsets of X closed under $\cap$ and $-$ contains the empty set $\emptyset$ and the full underlying set X, so we can also drop them in the list. It is easy to verify that in all fields of sets the following laws are valid.

$$
(1.29)\quad
\begin{array}{lcllcl}
S \cap (T \cap U) & = & (S \cap T) \cap U & S \cup (T \cap U) & = & (S \cup T) \cup U \\
S \cap T & = & T \cap S & S \cup T & = & T \cup S \\
S \cap S & = & S & S \cup S & = & S \\
(S \cap T) \cup S & = & S & (S \cup T) \cap S & = & S \\
(S \cap T) \cup U & = & (S \cup U) \cap (T \cup U) & (S \cup T) \cap U & = & (S \cap U) \cup (T \cap U) \\
-(S \cap T) & = & -S \cup -T & -(S \cup T) & = & -S \cap -T \\
S \cap -S & = & \emptyset & S \cup -S & = & X \\
S \cap \emptyset & = & \emptyset & S \cup \emptyset & = & S \\
S \cap X & = & S & S \cup X & = & X \\
- - S & = & S & & &
\end{array}
$$

Abstracting away from sets, a boolean algebra is any structure $\mathfrak{A} = \langle A, \cap, \cup, -, 0, 1\rangle$ where A is a set, $0, 1 \in A$, $-$ a unary operation on A and $\cap$ and $\cup$ binary operations on A, such that the laws above are valid for all elements of A, where $\emptyset$ is read as 0 and X as 1. It is a central result of boolean algebra that every boolean algebra can be construed as a field of sets, so the reader is invited to picture boolean algebras just as fields of sets whenever necessary. (This result is proved in the next section.) In a boolean algebra we write $x \leq y$ for $x \cup y = y$. If $x \cup y = y$, then also $x \cap y = x \cap (x \cup y) = x$, so we can characterize $x \leq y$ alternatively by $x \cap y = x$. If $x \leq y$ as well as $y \leq x$ we have $x = y$. For then $x \cup y = y$ from the first assumption and $x \cup y = x$ from the second. If $x \leq y$ then $(-y) \leq (-x)$, because $x \cup y = y$ can be transformed into $-(x \cap y) = -y$ and so $-x \cup -y = -y$, which is to say $-y \leq -x$. An element y is called the *complement* of x if $x \cap y = 0$ and $x \cup y = 1$. We show that $y = -x$, so that this is a characterization of $-x$, one which is used quite often. It is demonstrated as follows. First of all, we have from $x \cap y = 0$ that $x = x \cap 1 = x \cap (y \cup -y) = (x \cap y) \cup (x \cap (-y)) = 0 \cup x \cap (-y) = x \cap (-y)$, which means that $x \leq -y$. Second, from $x = x \cup 0 = x \cup (y \cap -y) = (x \cup y) \cap (x \cup -y) = 1 \cap (x \cup (-y)) = x \cup (-y)$ we deduce that $-y \leq x$. Hence, $-y \leq x \leq -y$, and so the two are equal. Moreover, if $x = -y$ then also $-x = - - y = y$.

Now let us discuss some concrete cases. Consider the set $\mathfrak{S}$ of all possible linguistic sounds. Every language cuts $\mathfrak{S}$ into different parts, according to its own system of distinctions. Let us be simplistic in assuming that any member of $\mathfrak{S}$ is identified by the speakers of a language as some sound belonging to his language. Indeed, a structuralist must make this assumption as long as he is not willing to concede that there are substantial i. e. phonetic properties that allow to identify sounds irrespective of the distinctive features operative in the language. This is a wide field, and we refrain from speculations here. Our previous analysis yielded certain distinctive features, such as *voiced* or *aspirated* and so on. A distinctive feature divides $\mathfrak{S}$ into two sets, namely one set to which all sounds belong which are positively specified for that feature, and the other set to which belong all sounds for which the feature is specified negatively. These sets are relative complements of each other. This means that the feature *voiced* selects the set $V \subseteq \mathfrak{S}$ of voiced sounds, while the set of voiceless sounds is just $\mathfrak{S} - V$, the relative complement. Given one distinctive feature, these are all the sets we can define with the distinctive features. Now take another feature, say *aspirated*. It defines the set A of aspirated sounds, and the set $\mathfrak{S} - A$ of nonaspirated sounds. With these two features we can define four basic sets, $F_{++} = V \cap A$, $F_{+-} = V \cap (-A)$, $F_{-+} = (-V) \cap A$ and $F_{--} = (-V) \cap (-A)$. There is a way to make precise the sense in which these sets are basic.

Definition 1.4.1 *Let $\mathfrak{A} = \langle A, \cap, \cup, -, 0, 1 \rangle$ be a boolean algebra and $B \subseteq A$. If B is closed under the opperations, we put $\mathfrak{B} = \langle B, \cap, \cup, -, 0, 1 \rangle$ and call it a* **subalgebra** *of $\mathfrak{A}$. Given an arbitrary set X the boolean algebra* **generated by** *X is the subalgebra defined by the smallest set B containing X being closed under the operations. We write $\langle X \rangle$ for B.*

This definition is successful because if X is a set and B_i an arbitrary collection of sets, all closed under the operations and containing X, then the intersection $\bigcap_{i \in I} B_i$ is another such collection. It is thus guaranteed that there is a smallest subalgebra containing X. Let us now ask, what is the structure of the field of sets generated by some set of features? In the case considered above, it is the algebra of sets which are representable by a union of the four sets F_{++}, F_{+-}, F_{-+} and F_{--}. There are $2^4 = 16$ such sets, at most. Notice that some of these set may be empty. For example, in most languages there is no sound which is both labial and dental (labiodentals are a different matter). Thus, the intersection of the set of dental sounds with the set of labial sounds is empty. But abstracting away from the specific context, it is quite conceivable that the intersection of all of the sets defined by the basic features is not empty. Such is the case with the sets *voiced* and *aspirated* features in Sanskrit. As our first theorem we will now prove the theorem of disjunctive normal form, which tells us how the algebra generated by a set X can be computed.

Proposition 1.4.2 (Disjunctive Normal Form) *Let $\mathfrak{A}$ be a boolean algebra and $E \subseteq A$. Assume that E is finite and $E = \{e_1, \dots, e_n\}$. For any subset $K \subseteq \{1, 2, \dots, n\}$ put*

(1.30) $\quad h_K = \bigcap_{i \in K} e_i \cap \bigcap_{i \notin K} -e_i.$

Then $\langle E \rangle$ consists of all unions of sets h_K. For infinite $X \subseteq A$,

(1.31) $\quad \langle X \rangle = \bigcup \{\langle E \rangle | E \subseteq X, E \textit{ finite}\}.$

Proof. To prove this claim it is only necessary to verify that the set thus defined contains E and is closed under the operations. That it is the smallest algebra follows from the fact that the elements defined above must certainly be in $\langle E \rangle$ by closure under the operations. Now, let us also notice that we need only show closure under intersection and complement. Namely, union is definable by $x \cup y = -(-x \cap -y)$. The bottom element 0 is the empty union, and 1 is its complement. For closure under intersection let us observe first that if $x = h_{K(1)} \cup h_{K(2)} \cup \dots h_{K(m)}$ and $y = h_{L(1)} \cup h_{L(2)} \cup \dots h_{L(n)}$ then

(1.32) $\quad x \cap y = \bigcup_{i \leq m, j \leq n} h_{K(i)} \cap h_{K(j)}.$

This is the generalized distributivity law. It holds for sets, and can be derived from the equations above. We are obviously done if we can show that the sets $h_{K(i)} \cap h_{L(j)}$ are of the type h_G for some G. Let us distinguish two cases. **Case 1.** $K(i) = L(j)$. Then the claim follows, for then $G = K(i) = L(j)$. **Claim 2.** $K(i) \neq L(j)$. Then there is a ℓ which is contained in one set, say $K(i)$, but not the other. This means that $K(i)$ is a conjunction containing e_ℓ, while $L(j)$ is a conjunction containing $-e_\ell$. Then $h_{K(i)} \cap h_{L(j)} = 0$, and so can dropped from the union. Finally, closure under complementation. If $x = h_{K(1)} \cup h_{K(2)} \cup \dots h_{K(m)}$ then we claim that

(1.33) $\quad -x = \bigcup_{N \neq K(1), \dots, K(m)} h_N.$

To see this, note that $1 = e_i \cup -e_i$ for all i, so that

(1.34) $\quad 1 = \bigcup_{K \subseteq \{1, \dots n\}} h_K.$

Thus, the union of x and $-x$ as defined is 1. The intersection is 0, and this uniquely characterizes the complement. To see that (1.31) holds, just observe that any term that can be formed from elements of X is formed by using finitely many elements only, so it is already in the right hand side. So the right hand side is included in the left hand side; the other inclusion is equally easy. $\dashv$

In the case of distinctive features we can conclude that the sets F_{++}, F_{+-}, F_{-+} and F_{--} defined above are of the type h_K for some $K \subseteq \{1, 2\}$. Any other set is a (possibly empty) union of these sets. They correspond to the set of sounds corresponding to a *phoneme*. Now what is the *phoneme* if these sets are just the realizations of a phoneme? We will not attempt an answer in a linguistical or philosophical sense. Rather, we want to point out that the field of subsets of $\mathfrak{S}$ induced by the features contains at most sixteen elements in the case of two features, and can be reconstructed over a much smaller set. Namely, we can choose for each set h_K a representative, for example simply K itself. An arbitrary element of the algebra, being a union of the h_K, is then simply the set of the representatives of those h_K participating in the union. In this way any element corresponds to a set $\{K_1, K_2, \dots, K_m\}$, where $K_i \subseteq \{1, 2, \dots, n\}$. In other words, we have reconstructed the algebra as the algebra of subsets of $\{1, 2, \dots, n\}$. A phonologist would use a different terminology here. He will name the representatives of the h_K *phonemes*. A *realization* of a phoneme is any sound that is being classified as belonging to the set for which the phoneme is the representative. Thus, $/p/$ is the representative of the class of all sounds classified (say by an English speaker) as 'p'. It is also the class of all sounds being classified by a phonologist as being voiceless, labial, aspirated etc. Any sound, that is, any member of $\mathfrak{S}$, is uniquely identifiable as being the realization of a phoneme. Crucially, as we have sets of sounds, we also have sets of phonemes, which have to be interpreted disjunctively. The set of all sounds realizing $\{/p/, /b/\}$ is the union of the sounds realizing $/p/$ and the sounds realizing $/b/$. In the present case phonologists speak of an *archiphoneme*. Archiphonemes cannot simply be equated with sets of sounds, though that would be a formally satisfying terminology. Phonologists restrict them to certain situations in which elements are not specified for a feature. (They correspond more closely perhaps to rectangles in the tensor product – for a definition see below.)

Sets allow natural operations, such as union, and so do boolean algebras. We will introduce these operations here with hindsight, assuming that boolean algebras are fields of sets. First, consider $\mathfrak{A}$, a field of sets over the set X, and $\mathfrak{B}$, a field of sets over Y. The disjoint union of X and Y, $X + Y$, is the smallest set into which X and Y can be embedded in such a way that no member of X is identical after embedding to a member of Y. This is a somewhat roundabout definition but there is no other way to define this notion. Consider the sets $\{1, 2, 3\}$ and $\{2, 3, 4\}$. Their union is $\{1, 2, 3, 4\}$. The disjoint union is something like $\{1, 2_X, 3_X, 2_Y, 3_Y, 4\}$. It has $3 + 3 = 6$ elements rather than four. The idea behind the disjoint union is that we want to distinguish the elements coming from X from those coming from Y regardless of whether they are actually identical. This causes a problem of identification of the elements of X in the disjoint union, and likewise for Y, which is solved by simply assuming that the disjoint union is not only a set but a set Z together with two embeddings, $\iota_X : X \to Z$, and $\iota_Y \to Z$. (Standardly, authors make sets disjoint by taking $X + Y$ to consist of elements $\langle 0, x\rangle$ for $x \in X$ and $\langle 1, y\rangle$ for $y \in Y$.) Standardly, one can view the disjoint union as a normal union where the elements of one of the sets are replaced by suitable different elements so as to make the two sets disjoint.

Now let us return to $\mathfrak{A}$ and $\mathfrak{B}$. We can actually rename the elements of Y in some way as to make X and Y disjoint. Then the standard union is isomorphic to the disjoint unions (i. e. they yield sets which can be mapped bijectively onto each other). We can now consider a new algebra over the set $X + Y$, as follows. Let C be the set of all unions of the form $S \cup T$, where $S \in A$ and $T \in B$. This is a boolean algebra. For it contains the empty set, being the union of $\emptyset_X \cup \emptyset_Y$, and it contains with each set $S \cup T$ the complement $(-S) \cup (-T)$. Moreover, C is closed under union, which is easy to demonstrate. Thus we have a boolean algebra, and it is called the *product* of $\mathfrak{A}$ and $\mathfrak{B}$. It is denoted by $\mathfrak{A} \times \mathfrak{B}$. We leave it to the reader to verify the following theorem.

Theorem 1.4.3 $\mathit{2}^X \times \mathit{2}^Y = \mathit{2}^{X+Y}$.

Now consider instead of the disjoint sum the set $X \times Y$ of pairs $\langle x, y\rangle$ such that $x \in X$ and $y \in Y$. With $S \in A$ and $T \in B$ we write $S \otimes T$ for the set $\{\langle x, y\rangle | x \in S, y \in T\}$ and call it a *rectangle* or a *basic tensor*. Under the set–interpretation a basic tensor is nothing but the product $S \times T$. Then let C be the set of all unions of basic tensors. We claim that it is a boolean algebra. Namely, $\emptyset \otimes T$ is the empty set, and $X \otimes Y$ is the full set. The intersection is easy to compute.

(1.35) $(S_1 \otimes T_1) \cap (S_2 \otimes T_2) = (S_1 \cap S_2) \otimes (T_1 \cap T_2)$

For a union of basic tensors, the intersection is then uniquely defined, and equal to a union of tensors. Finally, the complement is

(1.36) $-(S \otimes T) = (-S) \otimes T_1 \cup S_1 \otimes (-T)$

Thus we have successfully defined the algebra $\mathfrak{C}$ over $X \times Y$. We call it the *tensor product* of $\mathfrak{A}$ and $\mathfrak{B}$ and write $\mathfrak{A} \otimes \mathfrak{B}$. Again we leave it to the reader to verify that if $\mathfrak{A}$ is the set of subsets of X and $\mathfrak{B}$ the full set of subsets of Y, X and Y finite, then $\mathfrak{A} \otimes \mathfrak{B}$ is nothing but the algebra $\mathbf{2}^{X \times Y}$.

Theorem 1.4.4 $\mathit{2}^X \otimes \mathit{2}^Y = \mathit{2}^{X \times Y}$.

The tensor product arises naturally in phonemic analysis. Consider the case of a single distinctive feature, say *voiced*. We have seen that it divides the set of sounds into V and $-V$. Both are (up to now) phonemes. Call them $\mathfrak{v}$ and $\mathfrak{l}$. Consider now another distinctive feature, *aspirated*. Again, on its own it defines two phonemes, $\mathfrak{a}$ and $\mathfrak{p}$, say, realized by the sets A and $-A$. If the two are taken together, all four phonemes turn into archiphonemes, because each of them is underspecified for some feature. The new phonemes are now $\mathfrak{v} \otimes \mathfrak{a}$, $\mathfrak{v} \otimes \mathfrak{p}$, $\mathfrak{l} \otimes \mathfrak{a}$ and $\mathfrak{l} \otimes \mathfrak{p}$. So, given one distinctive feature we get 2 phonemes, with 2 distinctive features 4, and in general with n distinctive features 2^n phonemes (assuming that all combinations are realized, see also next section). The algebra of phonemes can thus be decomposed into the direct tensor product of the algebra of archiphonemes. In the extreme case, each feature defines its own archiphoneme, and the real phoneme is the tensor product of these archiphonemes. Notice also that while the tensor product lives on the product of the sets of the factors, there is no decomposition of $\mathfrak{S}$ into a product that reflects this. To make this clear, consider the archiphoneme $\mathfrak{v}$ and the archiphoneme $\mathfrak{a}$. The realization of the basic tensor $\mathfrak{v} \otimes \mathfrak{a}$ is the intersection (!) of the realizations of $\mathfrak{v}$ and the realizations of $\mathfrak{a}$. However, the phonematic reconstruction of the space of sounds is different. Under the hypothesis of a single distinctive feature it cuts the space $\mathfrak{S}$ into two sets, and each set is given a phoneme as a representative. Under the hypothesis of two distinctive features the once homogeneous sets are once again divided, yielding a division into (at most) four sets. So while the phonematic space grows larger, the subdivisions get smaller. But there is no sense in which the intersection $V \cap A$ can be seen as a set product of A and V. This is a mere artefact of the construction of the phoneme algebra as a set algebra. Even though this fact is somewhat bewildering, it analyses correctly the mode of inquiry. On the one hand we have the *concrete sounds* and a division into sets corresponding to abstract sounds, on the other we have an array of distinctive features, some of which are operative in a given language, some are not. The distinctive features create their own space of possibilities and may be just as concrete, for example in the form of recognition patterns or articulation instructions. We can easily recognize that the features *voiced* and *labial* adress different parts of the articulatory organ, the vocal chords in one case and the lips in the other, and so we can agree that the division into these features in the phoneme space has a real counterpart, and is not abstract. Both phonemes and sounds are at the same time concrete entities from their own point of view and abstract from the viewpoint of the other.

1.5 Duality Theory for Boolean Algebras

A *homomorphism* from a boolean algebra $\mathfrak{A}$ into a boolean algebra $\mathfrak{B}$ is a map $h : A \to B$ which satisfies the following structure preservation conditions.

$$\begin{array}{lrcl} & h(0) & = & 0 \\ & h(1) & = & 1 \\ (1.37) & h(-x) & = & -h(x) \\ & h(x \cap y) & = & h(x) \cap h(y) \\ & h(x \cup y) & = & h(x) \cup h(y) \end{array}$$

If h is surjective, $\mathfrak{B}$ is said to be a *homomorphic image* of $\mathfrak{A}$, and we also write $h : \mathfrak{A} \twoheadrightarrow \mathfrak{B}$. If it is injective, $\mathfrak{A}$ is a *subalgebra* of $\mathfrak{B}$. We call $\mathfrak{A}$ and $\mathfrak{B}$ *isomorphic* if there exists a bijective homomorphism $h : \mathfrak{A} \to \mathfrak{B}$. We write $\mathfrak{A} \cong \mathfrak{B}$ if the two algebras are isomorphic. Since the operations on a boolean algebra are interdefinable these conditions are not independent. It is enough to verify the conditions for intersection, complement and 1, for example. Given a homomorphism, we define the kernel of the homomorphism $ker(h) = \{\langle x, y\rangle | h(x) = h(y)\}$. Kernels can be given an abstract characterization.

Definition 1.5.1 *A* ***congruence*** *on a boolean algebra is a binary relation θ which is an equivalence relation and commutes with the operations in the following way. If $x \,\theta\, y$ then* *also* $-x\,\theta\,-y$. *If* $x_1\,\theta\,y_1$ *and* $x_2\,\theta\,y_2$ *then* $x_1 \cap x_2\,\theta\,y_1 \cap y_2$ *and* $x_1 \cup x_2\,\theta\,y_1 \cup y_2$.

The kernel of a homomorphism is a congruence. For it is an equivalence relation, as is easily seen. Moreover, if $h(x) = h(y)$, then $h(-x) = -h(x) = -h(y) = h(-y)$ and if $h(x_1) = h(y_1)$ as well as $h(x_2) = h(y_2)$ then $h(x_1 \cap x_2) = h(x_1) \cap h(x_2) = h(y_1) \cap h(y_2) = h(y_1 \cap y_2)$ as well as $h(x_1 \cup x_2) = h(x_1) \cup h(x_2) = h(y_1) \cup h(y_2) = h(y_1 \cup y_2)$. Conversely, given a congruence, let us define $[x]\theta = \{y|x\,\theta\,y\}$ and let us call it the *θ-block* of x. It is then straightforward to show that the operations can be defined blockwise. For example, if $y \in [x]\theta$ then $-y \in [-x]\theta$, for $y \in [x]\theta$ is the same as $y\,\theta\,x$, from which follows $-y\,\theta\,-x$, which is the same as $-y \in [-x]\theta$. This justifies defining the block-complement of $[x]\theta$ to be $[-x]\theta$. Similarly we have that if $y_1 \in [x_1]\theta$ and $y_2 \in [x_2]\theta$ then $y_1 \cap y_2 \in [x_1 \cap x_2]\theta$ as well as $y_1 \cup y_2 \in [x_1 \cup x_2]\theta$. Thus we can define the operations on entire θ-blocks.

$$\begin{array}{lrcl}
 & 1 & = & [1]\theta \\
 & 0 & = & [0]\theta \\
(1.38) & -[x]\theta & = & [-x]\theta \\
 & [x_1]\theta \cap [x_2]\theta & = & [x_1 \cap x_2]\theta \\
 & [x_2]\theta \cup [x_2]\theta & = & [x_1 \cup x_2]\theta
\end{array}$$

Finally, $A/\theta = \{[x]\theta|x \in A\}$. Now put $\mathfrak{A}/\theta = \langle A/\theta, \cap, \cup, -, 0, 1\rangle$. It is routine to check that the map $h_\theta : x \mapsto [x]\theta$ is a homomorphism. We call $\mathfrak{A}/\theta$ a *factor algebra* or *quotient* of $\mathfrak{A}$ obtained by *factoring out* the congruence θ. It turns out that congruences on boolean algebras are characterized by the class $[1]\theta$ alone. Such a class is called a *filter* and can be abstractly defined as follows.

Definition 1.5.2 *A **filter** on a boolean algebra is a set F satisfying the following three conditions. (fi1)* $1 \in F$, *(fi≤) If* $x \in F$ *then* $x \cup y \in F$, *(fi∩) If* $x, y \in F$ *then also* $x \cap y \in F$.

(Standardly, one writes (fi≤) as follows. *If* $x \in F$ *and* $x \leq y$ *then* $y \in F$. However, any element $y \geq x$ is of the form $x \cup y$; namely, $y = y \cup x$.) The simplest kind of filter is the so-called *principal filter*. It is of the form $\uparrow x = \{y|y \geq x\}$ for some x. Since $1 \geq x$, (fi1) holds. Moreover, if $y \geq x$ then $y \cup x = y$ and so $(y \cup z) \cup x = y \cup z$, which shows $y \cup z \in \uparrow x$. Finally, for $y, z \geq x$ we have $y \cap z \geq x$, and so all postulates hold. In general, the least filter containing a given set X of elements is the set of all elements z such that $z \geq \bigcap E$ for some finite $E \subseteq X$. If X is finite, then $x := \bigcap X$ is defined, and the filter is nothing but the principal filter $\uparrow x$. If θ is congruence, $[1]\theta$ is a filter. For $1 \in [1]\theta$, and so (fi1) is fulfilled. If $x \in [1]\theta$ then $x \cup y \in [1]\theta \cap [y]\theta = [1 \cup y]\theta = [1]\theta$, showing (fi≤). Finally, if $x, y \in F$ then $x \cap y \in [1 \cap 1]\theta = [1]\theta$, showing the condition (fi∩) to be fulfilled. Conversely, let F be a filter. Then put $\theta_F = \{\langle x, y\rangle|x\,\Delta\,y \in F\}$, where $x\,\Delta\,y = (x \cap y) \cup ((-x) \cap (-y))$. To show that this is an equivalence relation is a somewhat longish verification. Notice that $x\,\Delta\,x = (x \cap x) \cup ((-x) \cap (-x)) = x \cup (-x) = 1 \in F$. Furthermore, $x\,\Delta\,y = y\,\Delta\,x$, as the reader may verify, and finally $x\,\Delta\,z \geq x\,\Delta\,y \cap y\,\Delta\,z$.

$$(1.39)\quad \begin{aligned} x \mathbin{\Delta} y. \cap .y \mathbin{\Delta} z &= [(x \cap y) \cup (-x \cap -y)] \cap [(y \cap z) \cup (-y \cap -z)] \\ &= (x \cap y \cap z) \cup (x \cap y \cap -y \cap -z) \cup \\ &\qquad \cup (-x \cap -y \cap y \cap z) \cup (-x \cap -y \cap -z) \\ &= (x \cap y \cap z) \cup (-x \cap -y \cap -z) \\ &\leq (x \cap z) \cup (-x \cap -z) \\ &= x \mathbin{\Delta} z \end{aligned}$$

Furthermore, $(-x) \mathbin{\Delta} (-y) = ((-x) \cap (-y)) \cup ((--x) \cap (--y)) = (x \cap y) \cup ((-x) \cap (-y)) = x \mathbin{\Delta} y$. Next, consider $x_1 \mathbin{\Delta} y, x_2 \mathbin{\Delta} y \in F$. Then $(x_1 \cap x_2) \mathbin{\Delta} y = (x_1 \cap x_2 \cap y) \cup ((-x_1 \cup -x_2) \cap y) \geq (x_1 \mathbin{\Delta} y) \cap (x_2 \mathbin{\Delta} y)$ as can be verified by multiplying out the latter. By assumption the last is in F, since F is closed under (fi$\cap$), and so the first is also in F since it is closed under (fi$\leq$).

Theorem 1.5.3 *The map $F \mapsto \theta_F$ defined by $\theta_F = \{\langle x, y\rangle | x \mathbin{\Delta} y \in F\}$ is a bijection from the set of all filters over $\mathfrak{A}$ onto the set of all congruences over $\mathfrak{A}$. Its inverse is given by $\theta \mapsto F_\theta = [1]\theta$.* ⊣

Crucially, filters are in one-to-one correspondence with quotients $\mathfrak{A}/\theta_F$, which we also denote by $\mathfrak{A}/F$. A filter is called *proper* if it is not the full underlying set. An *ultrafilter* is a proper filter which has no proper extension. Alternatively, one can characterize an ultrafilter as a filter U such that $\mathfrak{A}/U \cong \mathbf{2}$. Each filter is contained in an ultrafilter. This is usually proved with Zorn's Lemma, but we will an intuitive argument here, resting on the axiom of choice.

Lemma 1.5.4 *Every proper filter of a boolean algebra is contained in an ultrafilter.*

Proof. If F is not an ultrafilter, take an element x such that $x \notin F$ and also $-x \notin F$. Consider the least filter F^1 containing x. It consist of all elements z such that $z \geq y \cap x$ for some $y \in F$. We claim that it is a proper filter. For if not, then $-x \in F^1$ and so there exists a $y \in F$ such that $-x \geq y \cap x$. But this means $-x \cap y \cap x = y \cap x$ and so $y \cap x = 0$. However, $y \cap x = 0$ implies $y = y \cap (x \cup -x) = y \cap -x$, that is, $y \leq -x$. Since $y \in F$, we must have $-x \in F$, in contradiction to our assumption. Thus, F^1 is a proper filter. If it is not an ultrafilter, continue this construction. This yields a chain $F = F^0 \subsetneq F^1 \subsetneq F^2 \subsetneq \ldots$. If we do not get an ultrafilter in finitely many steps, we will put $F^\omega = \bigcup_{i \in \omega} F^i$. It is a matter of direct verification that the union over an ascending chain of filters is a filter. Namely, since $1 \in F$, we will have $1 \in F^\omega$. Moreover, if $x \in F^\omega$ then $x \in F^i$ for some i, and by the fact that F^i is a filter, $x \cup y \in F^i$ and so also F^ω. Finally, if $x, y \in F^\omega$ then $x \in F^i$ and $y \in F^j$ for some $i, j \in \omega$. Without loss of generality assume $i \leq j$. Then $x \in F^j$, since $F^i \subseteq F^j$. Hence by (fi$\cap$) for F^j we have $x \cap y \in F^j$, whence $x \cap y \in F^\omega$. Now we can continue with F^ω etc. Eventually, we will have exhausted A and obtained an ultrafilter. ⊣

Theorem 1.5.5 (Stone Representation) *Every boolean algebra is isomorphic to a field of sets.*

Proof. Let $\mathfrak{A}$ be a boolean alegbra, and let $U(\mathfrak{A})$ be the set of all ultrafilters over $\mathfrak{A}$. We define $\widehat{x} = \{U \in U(\mathfrak{A}) | x \in U\}$. This is a homomorphism. For $\widehat{1} = U(\mathfrak{A})$, $\widehat{x \cap y} = \{U \in U(\mathfrak{A}) | x \cap y \in U\} = \{U \in U(\mathfrak{A}) | x \in U\} \cap \{U \in U(\mathfrak{A}) | y \in U\} = \widehat{x} \cap \widehat{y}$. This equivalence holds because if $x \cap y \in U$ then also $x = x \cup (x \cap y) \in U$ and $y = y \cup (x \cap y) \in U$, and conversely by (fi$\cap$). Finally, in an ultrafilter, $x \notin U$ exactly if $-x \in U$. For if $x \notin U$ then $h_U(x) = 0$, and so $h_U(-x) = -h_U(x) = 1$, in other words $-x \in U$. And conversely. Thus $\widehat{-x} = \{U \in U(\mathfrak{A}) | -x \in U\} = \{U \in U(\mathfrak{A}) | x \notin U\} = -\widehat{x}$. The image of the homomorphism, that is, the set of sets of the form $\widehat{x}$ for $x \in A$, form a field of sets over $U(\mathfrak{A})$, which is a homomorphic image of $\mathfrak{A}$. It remains to be shown that this map is injective. To show that it is enough to show that if $x \neq y$ then there is an ultrafilter such that $x \in U$ but $y \notin U$ or $x \notin U$ but $y \in U$. To put that differently, if $x \neq y$ then there is an ultrafilter containing $-(x \Delta y)$. Since the latter is nonzero in case the two are not equal, we are done if we can show that every element is contained in at least one ultrafilter. This follows from the previous lemma. ⊣

A boolean algebra is said to be *freely generated by* $X \subseteq A$ if for every map $h : X \to B$ there exists a unique extension $\overline{h} : \mathfrak{A} \to \mathfrak{B}$. The rationale behind that definition is as follows. $\mathfrak{A}$ is generated by X if $\langle X \rangle = A$, in other words, the smallest subalgebra containing X is $\mathfrak{A}$ itself. Moreover, we want to say that the elements of X are completely independent, that is, no relation holds between them unless forced by the laws of boolean algebra. This is expressed by the condition involving homomorphisms. We will see that in a minute. Let us first show that the structure of free algebras is determined only by the cardinality of the set X and that for any cardinality of X there exists an algebra freely generated by X. For the first claim consider two sets X and Y of the same cardinality, and let $\mathfrak{A}$ be freely generated by X, while $\mathfrak{B}$ is freely generated by Y. By assumption there exists a map $u : X \to Y$ and a map $v : Y \to X$ such that $u \circ v = id_Y$, the identity on Y, and $v \circ u = id_X$, the identity on X. We then get by freeness of $\mathfrak{A}$ a homomorphism $\overline{u} : \mathfrak{A} \to \mathfrak{B}$ and a homomorphism $\overline{v} : \mathfrak{B} \to \mathfrak{A}$, by the freeness of $\mathfrak{B}$. We know that $\overline{v} \circ \overline{u}(x) = x$ for $x \in X$. Again by the freeness of $\mathfrak{A}$ there is at most one homomorphism extending the identity on X, and this must then be the identitity homomorphism $1_{\mathfrak{A}} : \mathfrak{A} \to \mathfrak{A}$. Thus we must have $\overline{v} \circ \overline{u} = 1_{\mathfrak{A}}$. Similarly, $\overline{u} \circ \overline{v} = 1_{\mathfrak{B}}$ is proved. It follows that both $\overline{u}$ and $\overline{v}$ are bijective, and $\mathfrak{A}$ and $\mathfrak{B}$ are isomorphic. Now for the existence of the free algebras. Take any set X; we assume for simplicity that X is finite. Then let $Y = \wp(X)$. We put $\mathfrak{Fr}_B(X) = \mathbf{2}^Y$. We claim that this is the algebra freely generated by X. To see that, let $h : X \to B$. Extend that to a map from Y by putting $\overline{h}(E) = \bigcap_{e \in E} h(e) \cap \bigcap_{e \notin E} -h(e)$. Finally, a set $M \subseteq Y$ is mapped onto $\overline{h}(M) = \bigcup_{E \in M} \overline{h}(E)$. This is well-defined and a homomorphism.

Every boolean algebra is the homomorphic image of a free algebra. To see that just take $X = A$. There exists a unique homomorphism $\mathfrak{Fr}_B(X) \twoheadrightarrow \mathfrak{A}$, by freeness of the first. Of course, this is a crude way to get $\mathfrak{A}$, but it nevertheless proves the point. In general, if an algebra is not itself freely generated by a set X, but still generated by X, then there exists a surjective map $\mathfrak{Fr}_B(X) \twoheadrightarrow \mathfrak{A}$. This map has a kernel θ, which in turn defines a filter. So we can present $\mathfrak{A}$ as a pair consisting of generators and a filter. On the other hand, a filter may be a large set and can itself be defined by some elements. We may simply consider a least specification of a filter in the form of a set of elements such that the smallest filter containing the set is the filter we were looking for.

Definition 1.5.6 *A* **presentation** *of a boolean algebra $\mathfrak{A}$ is a pair $\langle X, E\rangle$ such that $X \subseteq A$ and the natural homomorphism $\mathfrak{Fr}_B(X) \to \mathfrak{A}$ defined by the least filter containing E is an isomorphism.*

What we have shown is that every algebra has a presentation. Let us look into some examples. We have seen earlier that the features *labial* and *dental* are not independent, that is to say, that out of the four possibilities of feature-value assignments only three can get realized. The picture we have presented in the last section about the tensor product of phoneme algebras is too simple to be true. We always have to reckon with interdependencies between the features. We conclude from the theorems above that we can nevertheless present the algebra of realizable phonemes by naming the distinctive features and and naming some nontrivial equations that holds between them. The Sanskrit system (with the addition of retroflex consonants) is presented by

$$\begin{aligned}(1.40)\qquad &\langle\{\mathit{aspirated}, \mathit{voiced}, \mathit{dental}, \mathit{labial}, \mathit{retroflex}\},\\ &\quad [-\mathit{dental}] \cup [-\mathit{labial}],\\ &\quad [-\mathit{dental}] \cup [-\mathit{retroflex}],\\ &\quad [-\mathit{labial}] \cup [-\mathit{retroflex}]\rangle\end{aligned}$$

This means that we have five features which are independent except for the fact that a consonant which is dental cannot be labial and also not retroflex, and a consonant which is labial cannot be dental or retroflex, and a retroflex consonant cannot be labial not dental. This means that the three features are mutually exclusive. They give rise to four archiphonemes, $/P/$, $/T/$, $/\d{T}/$ and $/K/$, with the following feature value specification.

(1.41)

	labial	*dental*	*retroflex*
$/P/$	+	−	−
$/T/$	−	+	−
$/\d{T}/$	−	−	+
$/K/$	−	−	−

These are in fact all combinations. Each archiphoneme gives rise to four phonemes if the values for *aspirated* and *voiced* are being instantiated. Since there are no restrictions on the latter combinations, we can decompose the phoneme algebra into a tensor product $\mathfrak{Fr}_B(1) \otimes \mathfrak{Fr}_B(1) \otimes \mathfrak{A}$, where the first factor corresponds to the *voiced* feature, the second to the *aspirated* feature and the last to the place of articulation. The latter is actually isomorphic to $\mathfrak{Fr}_B(2)$, so it too can be analysed as a tensor product. However, this would require making sense of the binary distinctions that get introduced by the decomposition. We will come to that point below in connection with the attribute-value matrices. Notice also that nothing is in the way of introducing another feature *velar* together with restrictions governing its cooccurrence with other features.

Let us mention another case, the syntactic categories. According to standard X-bar-syntax, syntactic categories are specified for two things, the bare or lexical category and the level. There are three levels, numbered from 0 to 2. Among the lexical categories are the so-called major lexical categories, *noun*, *verb*, *adjective* and *preposition*. The two components can be combined freely yielding a total number of twelve major categories. The major lexical categories have been analyzed as feature bundles, using two basic features, namely *verbal* and *nominal*. Then we have the following assignment

(1.42)

	verbal	*nominal*
noun	−	+
adjective	+	+
verb	−	+
preposition	−	−

This analysis can be justified on many grounds. It has moreover been claimed that linguistic rules can only make use of natural classes, that is, basic tensors. So we can expect laws to force nouns and adjectives to pattern one way, and verbs and prepositions another, or prepositions one way and verbs, nouns and adjectives another. But there should be no law that groups nouns and verbs together or adjectives and prepositions. Now when we come to the levels, we can attempt a similar analysis in terms of features. Naturally, we would opt not to group level 0 with level 2, but rather to group level 0 and level 1 into *non-phrasal* and/or level 1 and level 2 into *nonlexical*. In the standard system, there is then no level corresonding to [+*phrasal*] ∩ [+*lexical*]. Nevertheless, under different assumptions on X-bar-syntax such as recently proposed by Chomsky, we may concede the possibility that an element is both phrasal and lexical; a case in point are clitics.

1.6 Attribute Value Matrices

Out of the feature matrices developed the so-called feature-structures or attribute-value-matrices. These were originally only succinct ways to write down the classification of an item. For example, the lexeme /poetarum/ can be classified as

$$(1.43)\qquad \begin{bmatrix} \textsc{cat} & : & \textit{noun} \\ \textsc{gender} & : & \textit{masc} \\ \textsc{number} & : & \textit{plural} \\ \textsc{case} & : & \textit{gen} \end{bmatrix}$$

Many more syntactic properties can be listed in this way. Notice that rather than writing [−*singular*] we write [NUMBER : *sing*]. The difference is that we consider singular the *value* of the predicate or attribute NUMBER, as if asking what number this lexeme has. The advantage is clear when we come to CASE, because we can have any number of cases and it is not clear whether an analysis in terms of distinctive carries us anywhere. Notice that we can use distinctive features only to discriminate a given set into two parts and not arbitrarily many. In Latin we have five cases (disregarding the vocative) and it is easy to state this using attribute-value matrices. We simply say that the attribute CASE can have one and only one value which is to be taken from the set

(1.44) $\{nom, acc, gen, dat, instr\}$.

If we were to posit a differentiation in terms of distinctive features we have to look harder. (Linguists will prefer that because it means that the analysis will potentially give deeper insight into the Latin case system.) We may for example group nominative and accusative together (calling it, say, *structural case*), and then set genitive apart (calling it *adnominal case*), while dative and instrumental may be called *semantic case* or something else. Within these groups, nominative is less oblique, say, than accusative, and dative less oblique than instrumental. [2] But such groupings are generally hard to establish and cast doubt on the possibility to use distinctive features throughout. However, an analysis in terms of attributes and values is straightforward, as in the case of gender and number.

Now while feature bundles can be interpreted as vectors over a field, this is not possible with attribute-value matrices. The reason is that the coordinate spaces are dissimilar. We have three genders in Latin, but five cases. Moreover, the values in each coordinate are incomparable. There is no sense in comparing different cases with different genders. That means, while we could equate the values of the features in a feature bundles because they were all either + or −, no such equality is forthcoming here. Attribute-value matrices are highly complex entities, and we will return to them at various places. Let us for now make matters simple. An attribute-value language consists of a set of *features* and a set of *values* or *atoms*. Features are divided into two groups, so-called *category-valued* and *atom-valued* features. An atom-valued feature is assigned a *range*. This is the set of atoms which are allowed to be values of the features. A *category* is now defined as follows. Any set containing pairs of the form (i) [FEAT : *val*], where FEAT is an atomic valued feature and *val* an atom in the range of FEAT, or (ii) [FEAT : *cat*] where FEAT is a category valued feature and *cat* a category. This definition is recursive, allowing to stack attribute-value structures inside each other. This is useful in many respects. For example it can be used to state that an element selects another element. We can write this down by introducing a feature SELECT which will take attribute value structures as values, allowing us to be quite specific about the nature of the selected structure. In HPSG this role is played by SUBCAT, which is different though in that in takes a sequence of lexeme categories as values. [3] Another category valued feature is the SLASH-feature of GPSG. It was used to code the fact

[2]I am not in a position to judge whether this is a sensible grouping and whether the terminology is sound. This is just an example, so I give no warranty here as to its correctness.

[3]The massive overlap in the terminology is rather disturbing. The word *category* is used in a wide variety of senses in linguistics and it is not possible to remedy that. The reader should be aware that there is now a distinction between category in the narrow sense instances of whic are *noun*, *adjective* etc. and categegories in the broader sense which correspond more or less to the results of the context-substitution test. The latter is much more fine grained.

that the element in question had a gap inside it. Thus, in

(1.45) *Which book can you afford to buy?*

to buy, although ordinarily expecting a noun phrase to its right, is analysed as a verb-phrase with an additional slash-feature roughly as follows

$$(1.46)\quad \begin{bmatrix} \textsc{cat} & : & \textit{transitive verb} \\ \textsc{level} & : & 2 \\ \textsc{vform} & : & \textit{nonfinite} \\ \textsc{slash} & : & \begin{bmatrix} \textsc{cat} & : & \textit{noun} \\ \textsc{case} & : & \textit{acc} \end{bmatrix} \end{bmatrix}$$

Thus transitive verbs can appear in different syntactic environments, depending on the value of their SLASH-feature. If it is instantiated to some category that the verb selects for, then we can omit that element (it will have to appear elsewhere to guarantee the correctness of the SLASH-introduction, but that is another matter.) However, we must also find a way to represent the 'normal' situation in which there is no gap. Superficially it looks right to say that a verb phrase with no gap inside has the feature [SLASH : $\bot$], where $\bot$ is equivalent to the boolean constant 0. (Generally, the attribute-value matrices are of type boolean.) But this is not the correct analysis. We want that no value for SLASH is appropriate, so that has to be expressed differently.

In addition to this there also has been the need to introduce hierarchies into the features, that is, specify which feature can occur rightfully in a category that is itself the value of a given feature. This need arises in particular when we pursue the idea of expressing linguistic generalizations. For example, it is clear that among the features GENDER and NUMBER there is a greater coherence than between say GENDER and CASE. This is manifest in the fact that verbs and other lexical items select complements with a specific case, but not with specific number or gender. On the other hand, pronouns when referring back to a given noun phrase have to agree in gender and number but not in case with their antecedent. Number agreement can be deduced from the following critical data.

(1.47) *The boss has fired John. He has been too stupid.*
(1.48) *The boss has fired John. *They have been too stupid.*
(1.49) *The boss has fired John and Paul. * He has been too stupid.*
(1.50) *The boss has fired John and Paul. They have been too stupid.*

To encapsulate this in the notation we can introduce a new feature, INDEX, whose value is a category with only the features GENDER and NUMBER assigned within it. This proposal is being developed within HPSG, where index in addition is also specified for PERSON, which it must be if we follow the line of argumentation presented here. HPSG solves the problem of appropriateness of features as values of a feature by *typing* the attribute-value structures. The reason that CASE cannot get assigned inside the value of the INDEX-feature is because this value is an index, and for indices the case features are inappropriate. Thus, in HPSG we have the additional notion of a *type* to be able to express inappropriateness. HPSG attribute value structures in HPSG use among other a HEAD-feature whole value is a type coresponding to the rough classification into *noun*, *verb* etc. and which can be further specified for CASE etc. The reason why this works is that features are appropriate for certain types and not for others. We can directly state that CASE is appropriate for nouns, but not for verbs. It is, however, also possible to state directly that features select categories which contain only values for certain features and not other, but the approach taken in HPSG is intuitively more satisfying.

Let us focus a bit more on the notion of appropriateness. Already in the definition of an attribute-value matrix alias category we have made use of appropriateness by stating that features can only have values for a feature which are in the range of that feature. So, to write

(1.51) [CASE : *neuter*]

is not simply to write something false, it is to write down a syntactically illegitimate category. The value for the feature CASE must be drawn from the list of cases; it can neither be a different atom nor a category. This notion of appropriateness is based on logical presupposition because over and beyond syntactic well-formedness it puts (from a formal point of view arbitrary, i. e. not a priori valid) syntactic requirements on the structures. To give another example, certain lexical categories are not marked for case, such as adverbs, complementizers, prepositions and verbs. (At least in English this is so.) There are two options that we have. The first is to say that case is actually inappropriate for these categories or that they simply do not exhibit any distinctions in case. By and large this issue is not resolved. It is not a priori clear that adverbs show no case distinctions. After all, they are singled out only with respect to their syntactic position. We know also that many Latin adverbs are actually instrumental or ablative forms of adjectives, so that in principle there is a right to say that adverbs have case. Still, within a language like English we can claim that adverbs and verbs have no case. This is a problem for HPSG as long as we assume that categories look roughly like the one given for Latin /poetarum/. For then we cannot express this via the standard appropriateness constraints. If CASE is independent of the feature CAT then we cannot state that for some categories case is obligatory, for others it is inappropriate. Instead, if we want to do that we have to put one inside the other. We can either put CAT inside CASE or conversely. There is only one canonical way to do that; for it makes little sense to call CASE the major feature whose value is a category defined for CAT. Moreover, it would not solve the problem at hand. Instead we want to say that CASE is chosen inside a category which is the value of CAT. To make this work properly, we need to make the value of CAT a special type, namely that of a *category*. Then we can go on to say that for the type *noun* CASE is appropriate, while for the type *adverb* it is inappropriate.

This development of feature structures led to a concentration on the notion of a *type*. Types are properties, so they are open to a boolean analysis. In effect, the formalism for attribute value matrices has been enhanced by boolean constructors, written $\sqcap$, $\sqcup$ and $\neg$. For example, it may be necessary at times to state a disjunction. In the lexicon we might state that an element is actually either an adjective or an adverb, a noun or a verb etc.

We might say that some prepositions in German such as *wegen* (*because of*) select their complement with either dative or genitive case. Both are legitimate, although the use of the dative, being now more popular is considered uneducated. Leaving these connotations aside, either case is grammatical. Likewise, we can express the alternation between these two sentences

(1.52) *John gave a book to Mary.*

(1.53) *John gave Mary a book.*

(so called *dative-shift*) by stating a disjunction between the two selection requirements. To express such facts we write, for example,

(1.54) $[\textsc{cat} : \mathit{noun} \sqcup \mathit{verb}]$

to say that the value of CAT is either *noun* or *verb*. It is not clear whether disjunction is just a sign of lack of knowledge of whether it is inherent in the lexical items. This corresponds to the question whether the archiphonemes have a real status or whether they are just artefacts of the classification. We will not try to resolve this issue. Instead, let us just pursue the formal idea of introducing the connectives into the language of attribute-value matrices. We will say that there is a type of expression called *boolean*, and that there are boolean constants $\bot$, $\top$, the unary function symbol $\neg$ and the two binary function symbols $\sqcap$ and $\sqcup$. The laws governing the behaviour of these symbols are exactly those that we have in boolean algebras, with $\bot$ read as 0, $\top$ as 1, $\neg$ as $-$, $\sqcap$ as $\cap$ and $\sqcup$ read as $\cup$. There are possibly many constants of type boolean, such as the cases, and in general all atoms. Laws can be stated either by restriction on types or as appropriateness conditions. For example, we can introduce a type called *case* and also have various constants, such as *nom*, *acc* etc. To get the right connection between these constants, we need to state among other the following laws for the Latin case system. (We use $x \sqsubseteq y$ to abbreviate $x \sqcup y = y$.)

$$(1.55)\quad \begin{array}{rcl} \mathit{case} & = & \mathit{nom} \sqcup \mathit{acc} \sqcup \mathit{gen} \sqcup \mathit{dat} \sqcup \mathit{instr} \\ \mathit{nom} & \sqsubseteq & \neg\mathit{acc} \sqcap \neg\mathit{gen} \sqcap \neg\mathit{dat} \sqcap \neg\mathit{instr} \\ \mathit{acc} & \sqsubseteq & \neg\mathit{nom} \sqcap \neg\mathit{gen} \sqcap \neg\mathit{dat} \sqcap \neg\mathit{instr} \end{array}$$

(Notice that in the second and third line there is no equality, since items can have no case.) This parallels exactly the idea of a presentation of boolean algebras. Finally, inappropriateness can be stated quite simply as follows

(1.56) $[\textsc{case} : \mathit{neuter}] \quad \equiv \quad \bot$

This says that neuter is not a value of CASE. To make this work properly, we need to assume the following general law of attribute value matrices.

(1.57) $[\textsc{feat} : \mathit{value}_1 \sqcup \mathit{value}_1] \quad \equiv \quad [\textsc{feat} : \mathit{value}_1] \sqcup [\textsc{feat} : \mathit{value}_2]$

This is the law of $\sqcup$-*distributivity* for features. This would allow to say that if we specify a case value which is the disjunction between *neuter* and *nominative* then we can restate this as a disjunction between the case value being neuter (which is generally false) and the case value being nominative. So it is the same as saying the the element in question has case value nominative. Notice that even though this might seem nonsensical it is absolutely necessary if we want to rephrase inappropriateness constraints as equations or axioms for attribute value matrices. Moreover, let us consider the following problem. Nothing so far prohibits the following specification

$$(1.58)\qquad \begin{bmatrix} \textsc{cat} & : & \mathit{noun} \\ \textsc{case} & : & \mathit{nom} \\ \textsc{case} & : & \mathit{acc} \end{bmatrix}$$

It is contradictory on the standard interpretation. Notice that the problem arises for all categories. To prevent this we need to postulate a second law for attribute value matrices, namely

$$(1.59)\qquad [\textsc{feat} : \mathit{value}_1 \sqcap \mathit{value}_1] \quad \equiv \quad [\textsc{feat} : \mathit{value}_1] \sqcap [\textsc{feat} : \mathit{value}_2]$$

This is the law of ⊓-*distribution* for features. Thus, a feature can be given essentially only one value. Let us close by solving the question of the SLASH-feature. If we want to state that an element has no gap inside it, we will have to say that for no value x, $[\textsc{slash} : x]$ is true. This can be restated as

$$(1.60)\qquad \neg[\textsc{slash} : \top].$$

The statement $[\textsc{feat} : \bot]$ will generally be considered inappropriate. This is expressed by a third general law of attribute value matrices, the law of *proper assignment*.

$$(1.61)\qquad (\bot\mathbf{-hom}) \qquad [\textsc{feat} : \bot] \quad \equiv \quad \bot$$

1.7 A Boolean View on Attribute-Value Matrices

The attribute-value matrices can be thought of as a different way to specify what we have specified with boolean algebras. Indeed, we will see below that theories of attribute-value matrices can be thought of just as presentations of boolean algebras. However, the language of AVMS is usually finite and yet admits an infinite number of AVMS contrary to the situation in boolean algebras. Consider just the language with a single feature CASE and some values, such as $\{\mathit{nom}, \mathit{acc}, \mathit{gen}, \mathit{dat}, \mathit{instr}\}$. Then we can not only form the legitimate object $[\textsc{case} : \mathit{nom}]$ but all sorts of apparent nonsense, like

$$(1.62)\qquad \begin{bmatrix} \textsc{case} & : & \mathit{acc} \\ \textsc{case} & : & \begin{bmatrix} \textsc{case} & : & \mathit{nom} \\ \textsc{case} & : & [\textsc{case} : \mathit{dat}] \end{bmatrix} \end{bmatrix}$$

Essentially, this shows partly the strength and the weakness of the language of AVMS. Its strength is that we can iterate the construction, but on the other hand we have to stop it from doing so whenever that is inappropriate. Let us note that it is not really clear why we should resort to the language of AVMS instead of treating case as a property of noun-like phrases. In Indo-European languages, case is simply a property of nominal lexemes (adjectives and nouns) and to write [CASE : *acc*] is to implicitly treat the *accusative* as the value of a function assigning to each element its case whenever appropriate. This is intuitively motivated by locutions such as *the case of this noun phrase is accusative*. On the other hand, treating attributes as functions from objects to properties is a somewhat roundabout way of saying the same thing. The problem is also that using this function raises problems that we did not have before. We have to say that the function is only partial, and that it may not be used in particular on its own output. Thus, to ask for the case of the case of a noun phrase is illegitimate. The 'object' *accusative* (being the value of the feature CASE) is not the right thing to ask for its case. Remember that HPSG solves this by using types, but a boolean property is still the most economical solution. Reasons for not adopting it come from elsewhere, and we will encounter such reasons later.

It should also be stressed that typical AVM-languages do not view the atoms as booleans. That is to say, in the syntax of this language *acc* is an incomplete expression; only [CASE : *acc*] is a boolean. This should be borne in mind in the subsequent discussion. If instead we adopt a hybrid account, allowing both *acc* and [CASE : *acc*] as booleans, we could compress the above AVM into the following form.

$$(1.63)\qquad \left[\textsc{case} : \left[\begin{array}{lcl} \multicolumn{3}{l}{acc} \\ \textsc{case} & : & \left[\begin{array}{lcl} \multicolumn{3}{l}{nom} \\ \textsc{case} & : & dat \end{array}\right] \end{array}\right]\right]$$

We have deliberately chosen the example of CASE. With the feature SELECT (an equivalent of SUBCAT which does not take lists as values) this structure actually makes sense.

Let us now investigate in detail the classificatory power introduced by the features. The idea is as follows. We take an arbitrary boolean algebra $\mathfrak{A}$ of atomic values and a unary operator $\heartsuit$. The latter is called a *name forming operator*. For each $x \in \mathfrak{A}$, $\heartsuit\ell$ is a well-formed expression. Thus, unless specified otherwise, $\heartsuit$ cannot be stacked and so $\heartsuit\heartsuit x$ is meaningless. $\heartsuit$ allows to create a new boolean algebra $\mathfrak{A}^{\heartsuit}$. It is the boolean algebra generated by all expressions $\heartsuit x$, $x \in \mathfrak{A}$. We assume for a start that names are different if the labels are, a property which justfies the terminolgy of *name forming operator*. Now suppose that the number of elements of $\mathfrak{L}$ is 2^n, then $\mathfrak{L}^{\heartsuit}$ has 2^n generators, 2^{2^n} atoms and $2^{2^{2^n}}$ elements. For example, if $\mathfrak{L}$ is one-generated it has 4 elements, and $\mathfrak{L}^{\heartsuit}$ has 4 generators, 16 atoms and $2^{16} = 65536$ elements! Thus the numbers increase quite drastically. But this is only so if there are no laws governing the behaviour of $\heartsuit$; we say then that $\heartsuit$ is *imperspicuous*. However, we may consider the following laws for $\heartsuit$.

$$(1.64)\qquad \begin{array}{llcl} \textbf{0-homomorphy.} & \heartsuit 0 & = & 0 \\ \cup\textbf{-distribution.} & \heartsuit(x \cup y) & = & \heartsuit x. \cup .\heartsuit y \\ \cap\textbf{-distribution.} & \heartsuit(x \cap y) & = & \heartsuit x. \cap .\heartsuit y \end{array}$$

Under a different spelling we have met these laws of distribution in the previous section. We distinguish three types of name forming operators: *imperspicuous operators*, satisfying no logical laws, *half perspicuous operators*, satisfying 0-hom and $\cup$-distribution and *perspicuous operators*, satisfying 0-hom and both $\cup$- and $\cap$-distribution. The distributivity properties reduce the size of the name-algebra $\mathfrak{A}^\heartsuit$. If $\heartsuit$ is half perspicuous then in the finite case $\heartsuit x$ is equivalent to the disjunction of $\heartsuit a$, where a is an atom below x. Hence, $\mathfrak{A}^\heartsuit$ has as many generators as there are atoms of $\mathfrak{A}$; if $\mathfrak{A}$ has 2^n elements, it has n atoms and so $\mathfrak{A}^\heartsuit$ has n generators, 2^n atoms and consequently 2^{2^n} elements. Finally, if $\heartsuit$ is perspicuous it turns out that $\mathfrak{A}$ and $\mathfrak{A}^\heartsuit$ are almost isomorphic.

Proposition 1.7.1 *Let $\heartsuit$ be perspicuous. Then $\mathfrak{L}^\heartsuit \cong 2 \times \mathfrak{L}$.*

Proof. Consider the map $h_\heartsuit : x \mapsto \heartsuit x : \mathfrak{A} \to \mathfrak{A}^\heartsuit$. This map is a homomorphism with respect to $\cap$ and $\cup$ since it satisfies both $\cup$- and $\cap$-distribution; moreover, $\heartsuit 0 \equiv 0$. But this does not imply that $h_\heartsuit$ is a boolean homomorphism, because it does not necessarily respect negation. Nevertheless, $\mathfrak{A}^\heartsuit$ is a boolean algebra, so so we can study it via its space of ultrafilters. We will first show that $\mathfrak{A}^\heartsuit$ contains one more ultrafilter than does $\mathfrak{A}$. This will then establish the theorem. Consider an ultrafilter $U \in U(\mathfrak{A}^\heartsuit)$. Since the elements $\heartsuit x$ generate the boolean algebra, a general element of $\mathfrak{A}^\heartsuit$ is the union of intersections of the form $\heartsuit x_1 \cap \heartsuit x_2 \ldots \cap \heartsuit x_m \cap -\heartsuit y_1 \cap -\heartsuit y_2 \ldots \cap -\heartsuit y_n$. We can replace the conjunction over the $\heartsuit x_i$ by $\heartsuit \bigcap x_i$ and the conjunction over the $-\heartsuit y_j$ by $-\heartsuit \bigcup y_j$. Moreover, if an element $\heartsuit x \cap -\heartsuit y$ is in U, then so are $\heartsuit x$ and $-\heartsuit y$. Hence, U is fully determined by the sets $X_U = \{x | \heartsuit x \in U\}$ and $Y_U = \{y | -\heartsuit y \in U\}$. These sets therefore have to be studied. First of all it is clear that $X_U \cap Y_U = \emptyset$. Moreover, since $\heartsuit x \cup -\heartsuit x \in U$ we must have $X_U \cup Y_U = A$. So, X_U alone determines U. X_U is upward closed in $\mathfrak{A}$, that is, it satisfies (fi$\leq$); for if $\heartsuit x \in U$ then $\heartsuit(x \cup y) = \heartsuit x \cup \heartsuit y \in U$. Moreover, it also satisfies (fi$\cap$); for if x and $y \in X_U$ then $\heartsuit(x \cap y) = \heartsuit x \cap \heartsuit y \in U$, hence $x \cap y \in X_U$. Hence either X_U is empty or it is a filter; for if it contains any element whatsoever, it also contains 1. Finally, if it is not empty it is an ultrafilter. For assume that $x \cup y \in X_U$. Then $\heartsuit(x \cup y) = \heartsuit x \cup \heartsuit y \in U$. Since U is an ultrafilter, $\heartsuit x \in U$ or $\heartsuit y \in U$, from which $x \in X_U$ or $y \in X_U$. Thus we have shown that $X_U = \emptyset$ or X_U is an ultrafilter of $\mathfrak{A}$. What we need to show is that the field of sets of $\mathfrak{A}^\heartsuit$ can be obtained by taking the product of the field of sets for $\mathfrak{A}$ with the powerset $\mathbf{2}$. What we have established is simply a decomposition of the base set $U(\mathfrak{A}^\heartsuit)$ into disjoint sets C and D, where C corresponds to the set of ultrafilters of $\mathfrak{A}$ and D contains a single element. What we need to show that C and D are of the form $\widehat{x}$ for some x. Now take $x = 0$. We have $\widehat{\heartsuit 0} = \{U \in U(\mathfrak{A}^\heartsuit) | \heartsuit 0 \in U\}$. The set $X_{\widehat{\heartsuit 0}} = \{y | \heartsuit y \in \widehat{\heartsuit 0}\}$ is $= \emptyset$. Hence $\widehat{\heartsuit 0}$ corresponds to the set D; its negation corresponds to C, and this had to be shown. ⊣

This extra atom looks like a little anomaly but it is in fact quite a welcome addition. Consider the case of SLASH discussed earlier. It allowed us to state that an element had a gap, or equivalently, that some element that it subcategorized for has been displaced. Was it not for this extra atom we could not state that the SLASH-feature need not be instantiated, that is, that no displacement has taken place. Another example is CASE. If we accept that there are elements that have no case rather than any case we must have the possibility to express this. Thus for adverbs we can state $\neg[\text{CASE} : \top]$ rather than saying that adverbs do not change whatever their case is. Both have a plausibility. A perhaps clearer case is that of infinite verb forms. Typically, verbs inflect for person and number, but infinite verb forms do not. Although we may analyse this as saying that this is simply a lack of overt agreement, the explanation is usually that in the infinite forms the person and number features are absent, i. e. specifying them is inappropriate.

If, however, we insist that a feature must always be appropriate we must require the following.

(1.65) **1-homomorphy.** $\heartsuit 1 = 1$

Corollary 1.7.2 *Let* $\heartsuit$ *be perspicuous and satisfy 1-homomorphy. Then* $\mathfrak{L}^{\heartsuit} \equiv \mathfrak{L}$.

Proof. Indeed, now $C = \emptyset$ is exluded and so $h_{\heartsuit} : x \mapsto \heartsuit x$ is surjective. Hence it is bijective. We show now that $h_{\heartsuit}$ also respects negation. Namely, $\heartsuit x \cup \heartsuit - x. \equiv .\heartsuit(x \cup -x). \equiv .\heartsuit 1 \equiv 1$ as well as $\heartsuit x \cap \heartsuit - x. \equiv .\heartsuit(x \cap -x). \equiv .\heartsuit 0 \equiv 0$. Thus, $\heartsuit x$ is the complement of $\heartsuit - x$ showing that the map $x \mapsto \heartsuit x$ is a boolean homomorphism. ⊣

Finally, let us pursue the idea that both the atoms and the pair feature-value are booleans. This means effectively that we consider *acc* as a property of a lexical item as well as [CASE : *acc*]. This would be harmless, unless we want to allow for both forms to occur. In the case at hand we then have to say that both are equivocal. The postulate that ensures this is

(1.66) **Vacuity.** $\heartsuit x = x$.

All this taken together allows to compute the isomorphism type of the classification algebra obtainable from some atomic values and a given feature. Assume that the atomic values span an algebra $\mathfrak{A}$. Then allowing one iteration of the operator $\heartsuit$ we get the algebra $\mathfrak{A} \times \mathbf{2} \times \mathfrak{A} = \mathbf{2} \times \mathfrak{A}^2$. Allowing two iterations we get $\mathbf{2} \times \mathfrak{A}^2 \times \mathbf{2} \times \mathbf{2} \times \mathfrak{A}^2 = \mathbf{2}^3 \times \mathfrak{A}^4$ and so on. Consider now adding a new *feature*. What is the algebra of attribute value matrices with two features if the values are taken from a boolean algebra $\mathfrak{A}$? The answer is straightforward if we restrict to finite $\mathfrak{A}$. Then $\mathfrak{A}$ is atomic and so is $\mathfrak{A}^{\heartsuit}$ for any individual name forming operator. It has just one additional atom. Let us look at the atoms of the algebra with two name forming operators. Then the value of one feature is an atom of $\mathfrak{A}$ (if there is one) and the value of the second feature is an atom (again, if there is one). So the set of atoms is the product of the set of atoms of $\mathfrak{A}^{\heartsuit}$ with the set of atoms of $\mathfrak{A}^{\heartsuit}$. (Notice that we have two features, but the isomorphism type of the algebras is the same.) Hence the algebra is the tensor product of the two algebras.

Theorem 1.7.3 $\mathfrak{A}^{\heartsuit\spadesuit} \cong \mathfrak{A}^{\heartsuit} \otimes \mathfrak{A}^{\spadesuit}$. ⊣

This theorem holds even if $\mathfrak{A}$ is infinite. Thus, if only one iteration of the name forming operators is allowed, combining two sets of name forming operators corresponds to taking the tensor product of the name algebras. If more iterations are allowed, the result is of course more complex.

As with the features we can consider manipulating the set of atoms. For example, we can consider the case where a new value for an atom is being added (this makes sense diachronically). Abstractly put, we consider what happens if we take the names of $\mathfrak{A} \times \mathfrak{B}$, where $\mathfrak{A}$ and $\mathfrak{B}$ are classification algebras. We cane deduce easily the following theorem.

Proposition 1.7.4 $(\mathfrak{A} \times \mathfrak{B})^{\heartsuit} \cong \mathfrak{A} \times \mathfrak{B}^{\heartsuit} \cong \mathfrak{A}^{\heartsuit} \times \mathfrak{B}$. $\dashv$

This may be a puzzling result but it holds from a purely abstract point of view just because it tells us about the classificatory power of the system. For notice that if $\mathfrak{A}$ is an algebra, it corresponds to an intuitive classification. At the same time it may be isomorphic to another algebra $\mathfrak{B}$ classifying according to certain other properties. It is then possible to use $\mathfrak{B}$ instead of $\mathfrak{A}$ as the classificatory algebra. The labels that we use are just a help for humans so to speak, the formal system itself does not care what extrinsic meaning there is to the distinctions. This is the same situation as that of a phonologist who asserts that there is some difference between $/p/$ and $/b/$ but does not care what that property actually is. In the case above notice that have a name $\heartsuit$ corresponding to a feature and two classificatory algebras $\mathfrak{A}$ and $\mathfrak{B}$. Then $(\mathfrak{A} \times \mathfrak{B})^{\heartsuit} \cong \mathbf{2} \times \mathfrak{A} \times \mathfrak{B}$ because in addition to the possibility that the feature is inappropriate or that it takes a value from $\mathfrak{A}$ or a value from $\mathfrak{B}$. Now $\mathfrak{A}^{\heartsuit}$ allows for the first two, $\mathfrak{B}^{\heartsuit}$ for the first and the third. Thus we do not expect $(\mathfrak{A} \times \mathfrak{B})^{\heartsuit} \cong \mathfrak{A}^{\heartsuit} \times \mathfrak{B}^{\heartsuit}$ precisely because this would mean there are two ways in which the feature is inappropriate, either for values from $\mathfrak{A}$ or for values from $\mathfrak{B}$.

Chapter 2

Structure and Meaning

According to WILHELM VON HUMBOLDT language has an outer structure, which is visible in form of a string, and an inner structure, which is more complex. In this chapter we will show that the inner structure can in first approximation be assumed to be a tree. The argument will be based on the semantics of language and this will lead via MONTAGUE GRAMMAR to our first encounter of *categorial grammar*.

2.1 Strings versus Trees

Language is commonly understood to be a semiotic system with special properties. One of its most striking characteristics is for example the recursiveness, but there are more. Now, what exactly is a semiotic system? The answer must be vague, in fact part of the answer would supply an understanding of what language is, and so we must be content with a sketch. Crucial for the definition is the notion of a *sign*. FERDINAND DE SAUSSURE defined a sign as a pair consisting the material content of the sign, that by which the sign is identified qua sign. The other is the thing that the sign is taken to mean, again qua sign. DE SAUSSURE calles the first *signifiant*, that which means, and the second *signifié*, that which is meant. We can picture the sign *house* as follows.

(2.1) *house* $\begin{array}{|c|}\hline \text{/house/} \\ \hline \textbf{house}(x) \\ \hline \end{array}$

Here, /house/ is just an abstract phonemic representation, and **house**(x) an abstract semantic representation written in predicate logic. [1] The pairing itself is arbitrary. Thus the phoneme sequence corresponding to /house/ is the signifiant while the concept of a house

[1]We will oscillate between denoting the phonology of an item by writing it down as it would be spelled in writing, rather than noting the sequence of phonemes that would have to appear there. This saves having

is the signifié. We simply say that /house/ *means* 'house'. This sounds utterly simplistic, but is mainly due to the fact that we have to use the same words to denote the sign as well as its meaning. The arbitrariness of the pairing means that nothing permits us to deduce that this special sequence of phonemes means 'house'. In French the same meaning is paired with the string /maison/, in Finnish with /talo/ etc. Signs can be combined in a special way. We say that they *engage in a structure*. What this structure is is not a priori clear. What we can observe, however, is that both sounds and meanings can combine and in both cases we can find approximative answer as to how this structure looks like. We can say that sounds engage in a string. Although we will have to revise that assumption somewhat, it is more or less correct. For we observe that words are just put together in a linear fashion, one after the other. Moreover, there is no visible or audible extra structure, punctuation or intonation aside. Let us take that for granted. On the other hand, these strings have a meaning, too. What we are now interested in is to find out what structure we have to associate with meanings. This is a very difficult task, because we are not in a good position to say what the meaning of a given word is let alone the meaning of a sentence. Nevertheless, we can say that two sentences have the same meaning without being able to say what that meaning exactly is. As a guiding principle for the set-up of semantics, semanticists assume Frege's Principle. It says that the meaning of a complex entity is a function of the meaning of its proper parts. This needs some exegesis. We will assume that meanings engage in a structure. Then, according to this principle all that we need to find the meaning is just the individual meanings of the entities occurring in that structure plus the way in which they are embedded in the structure. Moreover, we expect that the function that yields this meaning is fixed for every type of structure. If the structure is a string, then we should expect by Frege's Principle that only the words and their position with respect to each other should matter.

Before we carry on, we will have to give some formal definitions. First, we need to clarify two fundamental concepts, that of a *string* and that of a *labelled tree*. Both have a presentation via a description and a presentation via a process that produces them. We call the first the *static presentation* and the other the *dynamic presentation*. Let us begin with strings. Under a static view, a string of elements of V is a function $f : \{1, 2, \dots, n\} \to V$. n is called the *length* of the string. The length may be 0, in which case the domain of the function is empty. This represents a legitimate element, the *empty string*. The function can assume the same value several times, in the extreme case in can just have one value. Thus, if we speak of $v \in V$ in the string we have to distinguish various *occurrences* of v in that string. We can do this by making reference to the number i such that $f(i)$ is that particular occurrence. We say that an occurrence of v *precedes* an occurrence of w if the first is $f(i)$ and the second is $f(j)$ with $i < j$. If $j = i + 1$ we say that (that occurrence of) v *immediately precedes* (that occurrence of) w. Alternatively, we can index the various

to elaborate on unnecessary details but also eliminates the need to use phonetic (!) transcription to refer to the phonemes. An accurate account would note the phonemes as bit vectors, but that complicates the matter beyond necessity. Similarly, much of what is declared to be the semantic of items is really just a shallow analysis using formal placeholders to avoid getting into detail.

occurrences of v in a string arbitrarily in order to make them distinct. Thus, rather than considering the range of f to be a set, we consider it to be a multiset. The best way to define a multiset over V in this connection is as a subset of $\omega \times V$. $x \in \omega \times V$ is called an *occurrence of* v. If $x = \langle i, v\rangle$ and $y = \langle j, v\rangle$ for some $v \in V$ and $i \neq j$, then x and y are called *different occurrences* of v. Now, a string can be characterized as a pair $\langle S, \sqsubset\rangle$ where $\sqsubset$ is a *linear order* on S, a set of occurrences of items from V. This means that the following holds.

$$\begin{array}{ll} & (\forall xyz)(x \sqsubset y \wedge y \sqsubset z. \rightarrow .x \sqsubset z) \\ (2.2) & (\forall x)\neg(x \sqsubset x) \\ & (\forall xy)(x \sqsubset y \vee x = y \vee y \sqsubset x) \end{array}$$

The first condition is referred to as the *transitivity* of the relation, the second as the *irreflexivity* and the third as the *linearity*. We write $x \sqsubseteq y$ if $x \sqsubset y$ or $x = y$. Notice that we are only dealing with *finite structures* here. In the infinite case there are many more restrictions that one might place on the relation(s), for example being discrete. But this does not arise here. We will show that the two definitions of a string are equivalent. First, let $f : \{1, 2, \ldots, n\} \rightarrow V$ be a function. Put $S = \{\langle v, i\rangle | f(i) = v\}$. Put $\langle v, i\rangle \sqsubset \langle w, j\rangle$ iff $i < j$. Then the three laws are satisfied by virtue of the fact that the natural numbers up to n satisfy them. Conversely, given a finite S and a linear order on S, there is a unique element x such that $x \sqsubseteq y$ for all $y \in S$. Let $f(1) = x$. Then consider $S - \{x\}$. The order $\sqsubset$ relativized to that set is a linear order, and so we have an element x' such that $x' \sqsubseteq y$ for all $y \in S - \{x\}$. Put $f(2) = x'$. And so on. This defines f. It can be shown that the procedures are up to isomophism inverses of each other. In some sense we can also say that a string is nothing but a *labelled linear order*. Namely, on the set of natural numbers $\{1, 2, \ldots, n\}$ we have a natural linear ordering. So the pair $\langle\{1, 2, \ldots, n\}, <\rangle$ is a linearly ordered set. The numbers will usually be referred to as *slots*. The function $f : \{1, 2, \ldots, n\} \rightarrow V$ is nothing but a labelling of the slots. Then we ordering on the occurrences of labels is the one that is imported from the natural numbers via f.

Now the third way to describe strings is the dynamic way. It is by using the binary connective $\cdot$, written as an infix operator. Whenever $\vec{x}$ and $\vec{y}$ are strings, so is $\vec{x} \cdot \vec{y}$. The set of strings is thus anything that can be obtained by concatenation. The associated function f can be built following the process of building the string in the following way. ϵ corresponds to the empty function. A single $v \in V$ corresponds to the function $f : \{1\} \rightarrow V : 1 \mapsto v$. Now, if $\vec{x}$ corresponds to the function $f : \{1, 2, \ldots, m\} \rightarrow V$ and $\vec{y}$ to the function $g : \{1, 2, \ldots, n\} \rightarrow V$, then $\vec{x} \cdot \vec{y}$ corresponds to the function $h : \{1, 2, \ldots, m{+}n\} \rightarrow V$ defined by $h(i) = f(i)$ if $i \leq m$ and $h(i) = g(i{-}m)$ if $m < i \leq m{+}n$. The binary symbol $\cdot$ represents a way to build a string by composing it from smaller units. However, different processes can yield the same string. That is to say, the dynamic presentation seems to present different objects where the static presentation can identify only one. For example, if we compose $\vec{x}$ with $\vec{y}$ and the result $\vec{x} \cdot \vec{y}$ with $\vec{z}$ then the resulting function is the same as if we had first composed $\vec{y}$ with $\vec{z}$ and then composed $\vec{x}$ with the result of the first operation. We say that $\cdot$ is an *associative* operation and by that we mean that it satisfies the following law.

$$(2.3)\qquad (\vec{x}\cdot\vec{y})\cdot\vec{z} = \vec{x}\cdot(\vec{y}\cdot\vec{z})$$

Given this fact, it can be shown that all that matters for the constitution of a string is only the respective order of the basic elements, not the particular history of combination. In other words, we can omit the brackets from the strings which helped us remember in which way it had been built. After that we can introduce any bracketing we like. All of them are equal according to the law above. Let us finally come to the notion of a *realization* of a string. We assume that the labels represent abstract phonemic representations, and that each if pronounced will give rise to an *utterance*. An utterance can simply be identified with what phoneticians look at when they study sounds. They use a spectral analysis of the utterance; this is a two dimensional picture, where the time is depicted along the x-axis and the frequency along the y-axis. This can in turn be construed as a function $u : I \to \mathfrak{W}$, where $I = [t_0, t_1[= \{t | t_0 \leq t < t_1\}$ is a half open interval in the set $\mathbb{R}$ of real numbers and $\mathfrak{W}$ is a suitable space for spectral analysis (for example, the set of reals or the set $\mathbb{Z}^{\mathbb{Z}}$ of functions from integers to integers, obtained by Fourier-transformation). [2] It is a crucial assumption that the domain of u is an *interval* and not any other set of real numbers. A function $u : [t_0, t_1[\to \mathfrak{W}$ is a realization of $\vec{x}\cdot\vec{y}$ iff there is a t_2 such that the restriction r_1 of u to $[t_0, t_2[$ is a realization of $\vec{x}$ and the restriction of u to $[t_2, t_1[$ is a realization of $\vec{y}$. This is in many ways overly simplistic. We mention a few points. First, it is known that the articulation of an abstract sound is different in different environments; moreover, it is sometimes impossible to segment the articulation of a sequence into a recognizable sequence of phonemes. This is due to the effect of *coarticulation*. We cannot say in a combination where one ends and the other begins, let alone assume that we can segment the sound in such a way that its individual parts are recognizable. In addition, speech consists also of *pauses*. These are meaningful elements; there is a difference between long and short pauses, although here as well it is next to impossible to say exactly what is long and what is short. We will not deal with these problems here. We will return to the problem of realization in connection with stratificationalism below.

The history of a string under composition can be visualized by a *tree*. Or conversely, a tree is nothing but a bracketed term. Statically, a tree is nothing but a pair $\langle T, <\rangle$ where the following holds

$$(2.4)\qquad \begin{array}{l} (\forall xyz)(x < y \wedge y < z. \to .x < z) \\ (\forall x)\neg(x < x) \\ (\forall xyz)(x \leq y, z. \to .y < z \vee y = z \vee z < y) \\ (\exists x)(\forall y)(y \leq x) \end{array}$$

The first two are known from linear orders. The third says that the relation is linear only if we restrict to the set of elements above a given element x. Let us write

[2]That the realization is a half open interval is just a technicality to make the definitions of overlap and precedence simple. Nothing changes substantially if we assume the interval to be closed – except of course the relevant definitions of precedence etc.

$$(2.5)\qquad \begin{aligned} \uparrow x &= \{y | x \leq y\} \\ \downarrow x &= \{y | y \leq x\} \end{aligned}$$

Then the third postulate says that $\uparrow x$ is always linear. In particular, linear orders are trees. The last postulate asserts the existence of a *root*. If it is not satisfied we call $\langle T, < \rangle$ a *forest*. Normally, we will mention the root explicitly, so that by a tree we normally understand a triple $\langle T, <, r \rangle$ where $T = \downarrow r$. Notice that r is unique. Furthermore, $\downarrow x$ is a tree for all x. This tree is called a *constituent*. The minimal elements with respect to $<$, that is, those elements x for which no y exists with $y < x$, are called *leaves*. These are the elements that constitute the tree. A *labelling* is a function assigning a label to each node. Notice that the labelling gives values to leaves and nonleaves. We will assume that there is a special set of labels reserved for leaves; these are the *terminal symbols*. The set of terminal symbols is the original vocabulary V. The other symbols are called *nonterminal symbols*. Thus a labelling is a function that assigns terminal symbols to leaves and nonterminal symbols to nonleaves.

It is our intention that the labelled tree should serve to uniquely define a string of terminal symbols. However, a tree does not specify an order on its leaves, though not all orderings will be compatible with a given tree. Thus we must specify an additional binary relation on T which we denote by $\sqsubset$. The guiding principle for its construction is the notion of a realization. We consider the tree as an elaborate structure over its leaves. Each constituent is a subtree, and it too should correspond to a string, in particular a substring. Here, a set of occurrences of labels is a *substring* of $\vec{w}$ if it is of the form $\{f(i), f(i+1), \dots, f(j)\}$ for some $1 \leq i \leq j \leq m$. This means that the tree defines a set of sets over its leaves and each is a substring. This is all we know. The relative order of the elements is still not defined. Although it would be enough to specify an ordering between the leaves, we will consider an ordering of the interior nodes as well. Namely, we extend the notion of a realization to constituents in the obvious way. We will say that the realization of a constituent $\downarrow x$ is the string of leaves contained in it. Each constituent x occupies a time segment $\tau(x)$, the *time* or *interval of utterance*. The function τ has the following properties.

$$(2.6)\qquad \begin{array}{ll} \tau(x) \subseteq \tau(y) & \text{if } x \leq y \\ \tau(x) \cap \tau(y) = \emptyset & \text{if } x \nleq y \nleq x \end{array}$$

Now define the relations *overlap*, $\circ$, by $x \circ y$ if $\tau(x) \cap \tau(y) \neq \emptyset$ and *precedence* $x \sqsubset y$ by (i) $\tau(x) \cap \tau(y) = \emptyset$ and (ii) all elements of $\tau(x)$ precede all elements of $\tau(y)$. Since $\tau(x)$ is an interval $[t_0, t_1[$ and $\tau(y)$ an interval $[u_0, u_1[$ we can rephrase $x \sqsubset y$ by $t_1 \leq u_0$ and $x \circ y$ by $t_1 \geq u_0$ and $u_1 \geq t_0$. This transfers the notions of overlap and precedence to the nodes of the tree. We can define these relations intrinsically as follows.

$$(2.7)\qquad \begin{array}{l} (\forall xy)(x \circ y. \leftrightarrow .x \leq y \vee y \leq x) \\ (\forall xyz)(x \sqsubset y \wedge z \leq y. \rightarrow .x \sqsubset z) \\ (\forall xyz)(x \sqsubset y \wedge z \leq x. \rightarrow .z \sqsubset y) \\ (\forall xyz)(x \sqsubset y \wedge y \sqsubset z. \rightarrow .x \sqsubset z) \\ (\forall xy)(\neg x \circ y. \leftrightarrow .x \sqsubset y \vee y \sqsubset x) \end{array}$$

(Again, note that we are only axiomatizing the finite structures.) Now, call a structure $\langle T, <, \circ, \sqsubset\rangle$ an *ordered tree* if $\langle T, <\rangle$ is a tree and $\sqsubset$ and $\circ$ satisfy the postulates (2.7).

Proposition 2.1.1 *Let $\langle T, <, \circ, \sqsubset\rangle$ be an ordered tree. Then the ordering $\sqsubset$ restricted to the leaves is a linear ordering.*

Proof. Let $L(T)$ be the set of leaves. $\sqsubset$ is transitive, so only irreflexivity and linearity have to be shown. If $x, y \in L(T)$ then either $x = y$ or $x \nleq y \nleq x$. In the first case we have $x \circ y$, in the second $\neg(x \circ y)$. Hence, if $x \neq y$ we have $x \sqsubset y$ or $y \sqsubset x$, by the last postulate of (2.26). This shows the linearity. Also, $x \sqsubset x$ cannot hold because $x \circ x$. ⊣

It can be shown that each linear order on the leaves gives rise to at most one ordered tree, as we will show below. However, there are linear orderings which cannot be seen as arising from an ordered tree under the given analysis of the string into a tree. The condition is precisely that the ordering must yield a substring of the given string for each constituent. Now let us be given a tree $\langle T, <\rangle$ and let $\sqsubset$ be a linear order over the leaves $L(T)$ of T. We call $\sqsubset$ *compatible* with $<$ if for every x, the set of terminals of $\downarrow x$ is a *convex* subset of $\langle L(T), <\rangle$, that is, a set of the form $[x, y] = \{z | x \leq z \leq y\}$.

Proposition 2.1.2 *Let $\mathfrak{T} = \langle T, <\rangle$ be a tree and $\sqsubset$ a linear ordering on the leaves of $\mathfrak{T}$. Then there is at most one way to define $\sqsubset$ and $\circ$ over $T \times T$ such that $\langle T, <, \circ, \sqsubset^+\rangle$ is an ordered tree and the ordering $\sqsubset^+$ restricted to the leaves is exactly $\sqsubset$. Such an ordering exists iff $\sqsubset$ is compatible with $<$.*

Proof. $x \circ y$ is defined by $x \leq y \vee y \leq x$. Define $\sqsubset^+$ as follows. $x \sqsubset^+ y$ iff $\neg(x \circ y)$ and there exists a leaf $d \leq x$ and a leaf $e \leq y$ such that $d \sqsubset e$. Notice that by the definition of an ordered tree, it holds that $x \sqsubset^+ y$ iff for some leaf $d \leq x$ and some leaf $e \leq y$ we have $d \sqsubset^+ e$. Since the latter is the same as $\sqsubset$, this is the only way to define $\sqsubset$. We first check whether this definition is sound. For that that it must be independent of the choice of d and e. Let $D = \downarrow x \cap L(T)$ and $E = \downarrow y \cap L(T)$. We have $d \in D$ and $e \in E$. What we need is that for all $d' \in D$ and $e' \in E$ we have $d' \sqsubset e'$. This is satisfied if $\sqsubset$ is compatible with $<$. For then D is a substring and E is a substring. Moreover, $D \cap E = \emptyset$ by the fact that $\neg(x \circ y)$. It is easy to see that all elements of D precede all elements of E. On the other hand, compatibility is also sufficient. For let there be a violation of compatibility. Then there exist x such that $\downarrow x \cap L(T)$ is not a substring. In particular, there are $d_1, d_2 \in D$ and $e_1 \notin D$ such that $d_1 \sqsubset e_1 \sqsubset d_2$. Then with respect to e_1 we have $x \sqsubset^+ e_1$ but also $e_1 \sqsubset^+ x$. This concludes the proof that the definition is sound iff $\sqsubset$ is compatible with $<$. If it is, it is routine to verify that with this definition the structure $\langle T, <, \circ, \sqsubset\rangle$ satisfies the requirements of an ordered tree. For example, assume $x \sqsubset^+ y$ and $z \leq y$. By the first there are $d \in \downarrow x \cap L(T)$ and $e \in \downarrow y \cap L(T)$ such that $d \sqsubset e$. Let $f \in \downarrow z \cap L(T)$. Since we have $\downarrow z \subseteq \downarrow y$, $f \in \downarrow y \cap L(T)$ and so $d \sqsubset f$ and hence $x \sqsubset^+ z$. ⊣

Another way to specify an ordering on the tree which has been used widely, e. g. in GPSG, is by specifying only the relations between the nodes immediately dominates by some node. This latter procedure has the advantage to be free of restrictions. To state the facts properly we will introduce some more terminology. We say that x *immediately dominates* y if $y < x$ but no z exists such that $y < z < x$. In that case we call x the *mother* of y and y the *daughter* of x. Also we write $y \prec x$ to state that y is a daughter of x. Notice that $\prec$ defines $<$ uniquely. Moreover, $<$ is the transitive closure of $\prec$. To characterize $\prec$ we use the notion of a local tree. A *local tree* is the subtree defined on x together with all its daughters. Thus a local tree is a very simple object consisting of elements $x, y_1, \dots, y_m$ where the only relation that holds is $y_i < x$. We call a tree where no three elements exist such that $x < y < z$ a tree of *height* 2. A tree can be seen as obtained by piecing together of all of its local subtrees. (We ignore the obvious exception that the tree consists of a single node.) Now given a set T and a set $\mathfrak{H}$ of trees of height 2 over subsets of T. We say that $\mathfrak{H}$ satisfies the *patchwork condition* if every element is either a mother or a daughter in some tree of $\mathfrak{H}$ and at most once a daughter in some tree of $\mathfrak{H}$. We say that $\mathfrak{H}$ satisfies the *root condition* if there is exactly one element that is a root in a tree but never a daughter. If $\mathfrak{H}$ is a collection of trees of height 2, put $x \prec y$ iff there exists a tree $\mathfrak{U} \in \mathfrak{H}$ such that $x < y$ in $\mathfrak{U}$. If $\mathfrak{H}$ satisfies the patchwork condition, for every x there can be at most one y such that $x \prec y$. This is enough to secure that $\mathfrak{T} = \langle T, < \rangle$ thus defined is a forest. To make it a tree we must assume that there is at most one element that is not a leaf in any tree of height 2. This is the *root condition*. Now consider the case of an ordered tree. A local subtree defines an ordering of the daughters of a node with respect to each other. So from a tree we get a collection of ordered trees of height 2. Conversely, let us be given such collection and let it satisfy both the patchwork and the root condition. Then we have a unique tree $\mathfrak{T} = \langle T, < \rangle$. Then put $x \sqsubset^+ y$ iff $\neg x \circ y$ and either (i) x and y are in the same local tree and $x \sqsubset y$ or (ii) there are z and w such that $x \leq z$ and $y \leq w$ and w, z are in the same local tree and $z \sqsubset w$.

Proposition 2.1.3 *A tree is defined uniquely by the set of all local subtrees. An ordered tree is defined uniquely by the set of all local ordered subtrees.* $\dashv$

Finally, let us turn to the dynamic presentation of a tree. Although the presentation can be given quite generally it is simpler if we assume that the trees are strictly binary branching. By that we mean that every mother has exactly two daughters. In that case the previous theorem tells us that the ordering on the tree is completely specified if we know just in what way the daughters of a node are ordered. Let us write $\oplus$ for the operation which takes two trees and forms a tree in which the two trees are immediate subtrees. This operation is called *merge*. Any binary branching tree can be presented as the result of an iterated merge. We start with the leaves $L(T)$ and define inductively a term for each node. For a leaf x let the term be just x itself. Then if the term is defined for x_1 and x_2 and x_1 and x_2 are the daughters of y, the term of y is $x_1 \oplus x_2$. The term that represents the tree is the term associated with the root.

2.2 The Algebra of Meanings

We have clarified in the previous section what we understand by such notions as string and trees. We want to present here arguments that show that unlike sounds, meanings do not form strings. Rather, they form trees as we will show. We will give first some abstract formulation of the problem; this will help us in understanding the role of *syntax*. In studying language we are interested among other in the map that pairs sound and meaning. Abstracting at the level of sounds we can say that we are interested in the mapping from sequences of phonemes into meaning(s). Abstracting still further we can ask about the map that mediates between strings of lexemes and meanings. It is the latter that we will look at now. Let us call m_L the map that assigns relative to the language L meanings to strings of words in L. It is a common assumption in linguistics that the meaning of a complex item should be determinable solely by the meaning of its basic parts and the mode of combination. This is yet loosely defined and will be made more rigorous later. It is called the *Frege-Principle*, after Gottlob Frege. If meanings engage in strings we will have, by Frege's Principle the equation

$$m_L(x \oplus y) = m_L(x) + m_L(y) \tag{2.8}$$

where $\oplus$ denotes the abstract sequencing operation for lexemes, and + the map that combines meanings. If $\oplus$ combines strings, then we would have the following equation

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z \tag{2.9}$$

This is simply the way strings behave. By Frege's Principle we then also have the equation

$$m_L(x) + (m_L(y) + m_L(z)) = (m_L(x) + m_L(y)) + m_L(z) \tag{2.10}$$

It is our aim to refute this equation of make it at least highly suspicious. This will then point to the fact that there is additional structure – which has to be postulated but whose existence is then motivated to some degree – such that (i) it allows to compute the output string of lexemes (ii) it allows to compute the meaning according to Frege's Principle. As a first result we will show that we have to assume a binary operation $\oplus$ which forms trees rather than strings out of the lexemes. It allows rather easily to reconstruct the sequence of lexemes. The difficult part is to show that it satisfies (ii). We cannot prove that really; rather, the argumentation will allow us to show in principle how *syntax* emerges. Syntax is the missing structure that we need to postulate in order to make (i) and (ii) hold. If it turns out that the trees are enough then we get the following result. There is a map s_L that associates strings with trees, and a map m_L that associates meanings. We have a map $\odot$, which we call the *Frege-function*. It is responsible for the combination of meanings. Furthermore, we have the following fundamental laws.

$$\begin{array}{rcl} s_L(x \oplus y) & = & s_L(x) + s_L(y) \\ m_L(x \oplus y) & = & m_L(x) \odot m_(y) \end{array} \tag{2.11}$$

These laws say that the maps m_L and s_L are homomorphisms from the structure algebra into the meaning algebra and the string algebra, respectively. The area of syntax is properly characterized as the search for the algebra of structure. The nature of the maps s_L and m_L is matter of the interaction of syntax with phonology and syntax and semantics. We will see that both are nontrivial.

Nor for the proof of the insufficiency of strings. Consider the following sentence.

(2.12) *Peter eats pizza.*

We want to argue that its meaning cannot simply be the meaning of /peter/ plus that of /eats/ and that of /pizza/. To refute that, however, it is not enough to point out that there is nothing corresponding to a string in semantics. What we have to show is that more is required to define the meaning than just the information that this string can provide. Let us make that precise. Look at the operation $\odot$, which 'strings' together pieces of meaning. If $\odot$ just strings the meanings together with no additional structure, then it must be associative. Then it would act roughly in the same way as conjunction. [3] That is to say, the model we would be proposing is that we allow each element to contribute a piece of information, so that /peter/ will be taken to mean (in informal rendering) *there is someone called Peter* and /eats/ to mean *there is someone who eats something* and finally /pizza/ to mean *there is pizza*. Then there is something that corresponds to the string, namely the lump of information that the three words provide. But it is easy to see that this lump is not the meaning we attribute to the sentence above. The problem is that when we say that /eats/ means *there is someone who eats something* or *someone eats something* there is no indication that the sentence *Peter eats pizza* tells us who that someone is, and what that something is. In other words, the verb supplies two roles, corresponding to argument places in logic, into which we may slot the objects supplied by the surrounding noun phrases. In standard truth conditional semantics we write the meaning of /eat/ as $\mathbf{eat}(x, y)$. Notice that what **eat** really means is a different question altogether. All the notation supplies is the fact that this word has two arguments with a specific interpretation in connection with an event of eating. [4] Typically, when there is an event of eating there is an *agent*, the one who eats, and a *theme*, that which is being eaten. All that matters here is that the agent is understood to fill the place of x and the theme the place of y. Now suppose that we interpret /peter/ as $\mathbf{peter}(x)$ and /pizza/ as $\mathbf{pizza}(x)$. Then the problem is to get at the following interpretation for the sentences.

[3]This is of course not a watertight argument. But notice that the conjunction *and* of natural language has the properties that we require. That is to say, it really acts like sequencing rather than boolean conjunction. This follows if we assume that to utter a certain sentence at time t we claim that it is true at that point of time. Metaphorically speaking ϕ gets time-stamped by uttering it. It is clear that uttering ϕ at t is a different claim than uttering the same sentence ϕ at time t'. Thus, ϕ *and* ϕ is not the same as ϕ, since to utter ϕ takes time and so ϕ may well cease to be true while we are uttering ϕ. By the same token ϕ *and* ψ is not the same as ψ *and* ϕ. However, *and* is associative if we assume that the items in a string receive the same time stamp regardless of the bracketing.

[4]This last paraphrase suggest that we are implicitly assuming an analysis in terms of events. Although there are many good arguments for that we will simply remain uncommitted here. Semantics will play a minor role throughout this work. The problem that is at hand is the same whatever reading we ultimately assume for the words of the language.

(2.13) $\mathbf{peter}(x) \wedge \mathbf{eat}(x, y) \wedge \mathbf{pizza}(y)$

It is normally assumed that the sentence means

(2.14) $(\exists x)(\exists y)(\mathbf{peter}(x) \wedge \mathbf{pizza}(y) \wedge \mathbf{eat}(x, y))$.

But this is a minor issue relating to the problem whether we assume the sentence to talk about entities or just assert a proposition. What interests us for the moment is the fact that we need to slot in the correct variables. Two solutions suggest themselves. The first method is to start off with

(2.15) $\mathbf{peter}(x) \odot \mathbf{eat}(y, z) \odot \mathbf{pizza}(w)$

This means that we insert the items in such a way that all variables in the distinct items get renamed prior to insertion in such a way that no variable in one item is equal to another variables in another item. Now, what we want to achieve is $\odot$ slots in identical variables for those variables which are identified as ranging over the same thing. To make that work we actually assume that we start off not with the original sentence but with the following annotated sentence.

(2.16) $\mathit{Peter}_1 \quad \mathit{eats}_{1,2} \quad \mathit{pizza}_2$.

The additional numbers are called *indices*. Let us first ignore the problem of how to get at the assignment of indices. Then instead of the above translation we get the following after insertion of items and renaming of variables.

(2.17) $\mathbf{peter}_1(x) \odot \mathbf{eat}_{1,2}(y, z) \odot \mathbf{pizza}_2(w)$.

The instruction for $\odot$ is to assign the variables to indices. This is straightforward, given that the sequence of variables is of equal length to the sequence of variables. In fact, in syntactic theory it is assumed that this assignment of variables to indices is taken care of in some explicit way by the lexical item. That is to say, it is not uniform, but the lexical item specifies how it is to be done. Now all variables get identified which bear the same index; we have for the index 1 the variables x and y and for the index 2 the variables z and w. $\odot$ renames the variables, one for each index, and finally changes to $\wedge$. In the end we get

(2.18) $\mathbf{peter}(x) \wedge \mathbf{eat}(x, y) \wedge \mathbf{pizza}(y)$.

This is the interpretation we claim it has. Now let us return to the problem of assigning the indices. We assume here that the verb, being the glue between the noun phrases, may choose an indexation of its variables. Let the indices be 1 and 2. We have 1 corresponding to the agent and 2 corresponding to the theme. Now in English the first noun phrase alias subject will carry the same index as the agent, and the second noun phrase alias direct object the same index as the theme. In Latin, the noun phrase bearing nominative case will (normally) be the agent, and the noun phrase bearing accusative case the theme. Thus everything is reduced to a proper regime of argument places.

The second solution has gained more widespread acceptance due to the work by Richard Montague. The reason is partly that it solved much more problems that just the one we are dealing with here. The basic assumption is that the sequencing operation is not associative. That is, we do not have

(2.19) $\alpha \odot (\beta \odot \gamma) = (\alpha \odot \beta) \odot \gamma.$

If that is so, then the structure meanings engage in is not that of a string, and this is the main point we arguing for here. MONTAGUE assumes that the Frege-function, that is, the function specifying the way in which meanings combine is not addition or conjunction or anything like that but just *function application*. To make that work, several adaptations have to be made. First, we need to assume that if two meanings are combined via the merge $\odot$ one is a function and the other is a suitable argument. Moreover, an element taking several arguments if read as a function must be rendered into a function that can take its arguments one at a time. The latter is referred to *Currying* a function, after HASKELL CURRY. These two changes can easily be achieved in the so-called *λ-calculus*. The idea is that any term with a free variable, such as $\mathbf{peter}(x)$ or $\mathbf{eat}(x, y)$ can be turned into a function f which will insert the argument exactly into those positions where the variable has been found before. The way to write f is as follows.

(2.20) $\lambda x.\mathbf{peter}(x)$

Notice that this is a function which can be applied to objects and yields an expression asserting that the object has the property of being called Peter. As in standard notation of mathematics, we write $f(c)$ for the result of applying the function f to the object c. Thus, if u is a certain individual (or the name of an individual, for that matter [5]) then $\lambda x.\mathbf{peter}(x)(u)$ stands for the result of plugging in u at those places where x has been. Hence it is identical to $\mathbf{peter}(u)$. Notice that the prefix λx has two ingredients; one is the so-called λ-abstractor and the other is the variable. The λ-abstractor turns the expression into a function over the variable. This process can be iterated. We may write

(2.21) $\lambda y \lambda x.\mathbf{eat}(x, y)$

which is a function eating one argument to yield a function that eats another argument. Notice that this is not the same function as

(2.22) $\lambda x \lambda y.\mathbf{eat}(x, y)$

The first is a function that assigns to the first argument (which is in fact its only argument – the other is visible only after applying the function to this argument) the theme, while the second function assigns the argument to the agent. In our somewhat fancy notation we have a difference between the brackets that constitute the term and the ones that derive from application. Assuming that the Frege-map is function composition $\lambda x.\mathbf{peter}(x) \odot u$ denotes the result of applying the function $\lambda x.\mathbf{peter}(x)$ to u. The result is $\mathbf{peter}(u)$ and is

[5]By that we mean that u is an individual constant, something which must be mapped onto one and only one object. Logical names are not like ordinary names, which may be given to many people at the same time. So, there is the property of being called Peter, while at the same time we may also speak of Peter *as if of an individual*. This is justified in contexts where this name will identify an individual by that name. Notice that language has almost no logical names. Each expression turn into a property at closer look. Possible exceptions are indexical expressions, such as *I*, *here* and *now*.

obtained by a process that is referred to as β-conversion. We write

(2.23) $\lambda x.f(x) \odot u \rhd f(u)$

We have now achieved two things. We have turned the verb into a function, and we have obtained a function that does not eat its arguments in one fell swoop, but one by one. What we have obtained now is precisely an instance of the nonassociativity of $\odot$, because it does matter in which order we supply the arguments. In other words, we have a difference between the following two structures.

(2.24) $(u \odot \lambda y \lambda x.f(x, y)) \odot v$
(2.25) $u \odot (\lambda y \lambda x.f(x, y) \odot v)$

The first reduces to $f(v, u)$ and the second to $f(u, v)$. Notice that $\odot$ does not care about order. So we have obtained a solution to the problem in λ-calculus which uses λ-terms to organize the association of indices with variables. However, it works on the fundamental assumption that the structure we work with is not a string, but a binary branching tree. Notice that in this particular example no further assumptions had to be made concerning the indexation or variable regime. The only thing that has to be looked out for (as in the previous case) is that prior to insertion, no variable occurs in distinct elements that constitute the leaves of the tree.

However, the solution as just sketched is not exactly the one proposed by Montague. There are many reasons for this. The first is that it will not work even in the present case. There is no way in which we can plug in the meaning of the noun phrases into the argument places of the verb at hand. Originally, Montague had no problem with that because he saw *Peter* as a logical name, and so standing for an individual. He was more concerned with expresssions such as *every man* which could not possibly be interpreted as names. For reasons of uniformity he then adopted for all noun phrases the interpretation that he chose for the quantified noun phrases. The philosophical reasoning he gave is of no immediate concern here. The interesting fact is that he saw that if we combine two elements, x and a function f taking x as an argument, we could also turn things around and call x the function and f its argument. Write $x(f)$ and define it by $x(f) := f(x)$. As it stands, this sound rather nonsensical because there seems to be a substantial difference between function and argument. However, this is only apparently so. Nothing prevents us to view x as the function and f as its argument. In fact, the only thing that prevented this was the *typing* regime imposed onto the λ-calculus to keep it from being inconsistent. It is assumed that each object has *type*, and that the basic types are *objects*, denoted by e, *worlds*, denoted by s, and truth values, denoted by t. Complex types are formed from these types by bracketing. Then we have (i) s, e and t are types, (ii) if α and β are types, so is $\langle\alpha, \beta\rangle$. A function is always of type $\langle\alpha, \beta\rangle$, where α is the type of its argument and β the type of its value. A predicate, for example, is a function of type $\langle e, t\rangle$ yielding a truth-value when given an object. A transitive verb is a function of type $\langle e, \langle e, t\rangle\rangle$. [6] Crucially,

[6]We are completely ignoring intensions here. This applies also to the Frege-map. If we were to admit intensions, we would have to split the Frege-map into several subcases, according to whether the argument

a quantified noun phrase is interpreted as a different function than an object. For reasons that shall become clear later we must assume that a subject noun phrase is to be construed as a function that yields a truth value when given an intransitive verb. Thus, a subject has the type $\langle\langle e,t\rangle,t\rangle$. A direct object, on the other hand, is then interpreted as a function yielding an intransitive verb when given a transitive verb. That means, it must be of type $\langle\langle e,\langle e,t\rangle\rangle,\langle e,t\rangle\rangle$. We summarize this in the following table.

(2.26)	/peter/	$\lambda\mathcal{P}\lambda x.\mathbf{peter}(x)\wedge\mathcal{P}(x)$	$\langle\langle e,t\rangle,t\rangle$
	/eats/	$\lambda y\lambda x.\mathbf{eat}(x,y)$	$\langle e,\langle e,t\rangle\rangle$
	/pizza/	$\lambda\mathcal{Q}\lambda x.\mathbf{pizza}(x)\wedge\mathcal{Q}(x)$	$\langle\langle e,\langle e,t\rangle\rangle,\langle e,t\rangle\rangle$

Here the variable $\mathcal{P}$ is of type $\langle e,t\rangle$ and the variable $\mathcal{Q}$ of type $\langle\langle e,t\rangle,t\rangle$. The typing regime introduced in this way allows to define the notions of direct object and subject purely by their respective types. In fact, it was MONTAGUES original conception that the syntax should follow directly from the type assignment. This gave a big impetus for the development of categorial grammar.

2.3 Constituents

Given a tree $\mathfrak{T}=\langle T,<\rangle$ we have defined a constituent to be a subtree consisting of the set $\downarrow x=\{y|y\leq x\}$ and the order induced by $\mathfrak{T}$ on it. An ordered tree gives rise to an ordered constituent. On the other hand, if we look at the strings over which the tree is built, a constituent is often also referred to by the set of leaves that the subtree $\downarrow x$ contains. There is a certain terminological clash here, but it is usually resolved by the context. A tree defines over its set of leaves a *constituent structure*, which consists of all sets of the form $\downarrow x\cap L(T)$. Constituent structure arises from the observation that certain words of a sentence form a group, that is, they are closer together than others. In the light of the previous discussion we can say that semantics introduces such a grouping into consituents. We will find out more about that later. Now consider the following sentence.

(2.27) *Peter reads books.*

Given the analysis of the previous section we get the tree analysis that leads to the following constituents.

(2.28) $\{\{\mathit{Peter}\},\{\mathit{reads}\},\{\mathit{books}\},\{\mathit{reads},\mathit{books}\},\{\mathit{Peter},\mathit{reads},\mathit{books}\}\}$

A *constituent structure* can also be intrinsically characterized as follows. A *constituent structure* is a pair $\langle C,\mathbb{C}\rangle$ where C is a set, and $\mathbb{C}\subseteq\wp(C)$ a set of subsets such that (i) $\{x\}\in\mathbb{C}$ for all $x\in C$, (ii) for all $D,E\in\mathbb{C}$ either $D\subseteq E$, $E\subseteq D$ or $D\cap E=\emptyset$ and (iii) $C\in\mathbb{C}$. An *ordered constituent structure* is a triple $\langle C,\sqsubset,\mathbb{C}\rangle$ such that (i) $\langle C,\mathbb{C}\rangle$ is a constituent structure, (ii) $\langle C,\sqsubset\rangle$ is a string and (iii) each $D\in\mathbb{C}$ is convex,

is intensional or extensional, and the result should be intensional or extensional. It is quite plausible however that a uniformity can be achieved by a more fine grained syntax. (See (**?**).)

that is, of the form $[x, y] = \{z | x \sqsubseteq z \sqsubseteq y\}$. A tree $\mathfrak{T}$ defines a constituent structure $\mathfrak{Cs}(\mathfrak{T}) = \langle L(T), \{\downarrow x \cap L(T) | x \in T\}\rangle$. In the same way an ordered tree defines an ordered constituent structure. A tree is *strictly branching* if every node has more than one daughter.

Proposition 2.3.1 *Let $\mathfrak{C}$ be a(n) (ordered) constituent structure. There are infinitely many trees whose constituent structure is $\mathfrak{T}$. However, there is exactly one strictly branching tree $\mathfrak{T}$ whose constituent structure is $\mathfrak{C}$.*

Proof. We show the second claim first. Let $\mathfrak{C}$ be a constituent structure. Let $T = \mathbb{C}$. Furthermore, put $C < D$ iff $C \subsetneq D$. Then $\langle T, <\rangle$ is a tree. For the ordering is irreflexive and transitive. Moreover, if $D_1, D_2 \supseteq C$. Then $D_1 \cap D_2 \neq \emptyset$ and so $D_1 \subsetneq D_2$ or $D_1 = D_2$ or $D_1 \supsetneq D_2$. Finally, every node in $\mathfrak{T}$ which is not branching is a leaf. Namely, consider a set C. Let C immediately dominate D. Then there is a $x \in C - D$. Let E be the largest set $\subsetneq C$ containing x. We claim that E is immediately dominated by C. For take any larger set $X \supsetneq E$. Then either $X \subsetneq C$, $X = C$ or $X \supsetneq C$. The first cannot hold by choice of E. The last two imply our claim. Finally, $E \cap D = \emptyset$ because $x \in E$ but $x \notin D$, so $E \neq D$. Since they are both immediately dominated by C, C is branching. Now let $\mathfrak{C}$ be ordered. Then there exists an ordering on $\mathfrak{T}$ defined by $\mathfrak{C}$ iff $\sqsubset$ is compatible with $<$, that is, iff constituents are convex sets. Now for the first claim. Take any node x in a tree $\langle T, <\rangle$ and replace it by two copies x_1 and x_2 as follows. Whenever $y < x$ put $y < x_1$ and $y < x_2$, and whenever $y > x$ put $y > x_1$ and $y > x_2$. Finally, put $x_1 < x_2$. This defines the tree $\mathfrak{T}^+$. The construction replaces x by two copies x_1 and x_2 such that x_1 is the only daughter of x_2. Assume that x was not a leaf; then neither x_1 nor x_2 is a leaf. From the construction it follows that $\downarrow x_1 \cap L(T^+) = \downarrow x_2 \cap L(T^+) = \downarrow x \cap L(T)$. For the remaining elements we also have $\downarrow y \cap L(T^+) = \downarrow y \cap L(T)$, and this shows that $\mathfrak{T}^+$ and $\mathfrak{T}$ have the same constituent structure. The construction can be iterated arbitrarily often. This shows that there are infinitely many trees with the same constituent structure. $\dashv$

Using standard set-theoretic constructions it is possible to code any tree into set theoretic notation. Namely, let $\mathfrak{T} = \langle T, <\rangle$ be given. We define by induction on the height of a node te function γ. A leaf x is mapped onto $\{x\}$. If γ is defined on all daughters $y_1, y_2, \dots, , y_m$ of x then

(2.29) $\quad \gamma(x) := \{\gamma(y_1), \gamma(y_2), \dots, \gamma(y_m)\}$

If we want to code the ordering, we have to use sequences rather than sets, that is, we have to put

(2.30) $\quad \sigma(x) := \langle \sigma(y_1), \sigma(y_2), \dots, \sigma(y_m)\rangle.$

This is the same as the usual bracket notation if instead of $\langle a, b, c\rangle$ we write $[a, b, c]$.

We have indicated earlier that the semantic analysis of Montague provided evidence that we have to assume a constituent structure for sentences. If we set aside meaning as criterial evidence we might ask whether there are intrinsic syntactical reasons for assuming constituents, and what may count as evidence. The facts are not as straightforward as with classification. There are many tests for detecting constituents, and they give rise to some strikingly different constituent analysis. It cannot expected that we will solve that problem by some magical definition. However, we will propose a number of criteria which are quite robust. The first definition is that anything is a constituent in a given context that can be replaced by a single lexical item in the same context. This presupposes that a constituent is a convex subpart of the string. The rationale for this test is as follows. If in a string something is a constituent, then it functions as a kind of complex lexeme. We expect that it should not make a difference syntactically whether something is a single lexeme or a string of lexemes as long as they are of the same type. This reasoning, however, is circular as long as types are defined for strings as in Chapter 1. Because then if the string $\vec{v}$ is basic, that is, has the same context set as x, then they are intersubstitutable in all contexts by definition. Thus, to avoid the apparent circle we need to define basic syntactic categories only for single lexemes. After that we define the category of a string to be that of a single lexeme if it has the same context set.

Definition 2.3.2 *Let L be a language over the vocabulary V. The set of* ***syntactic categories*** *of L is the set of all context sets $\mathfrak{C}_L(x)$ for $x \in V$. A* ***basic context*** *of L is a context $C = \langle \vec{w}_1, \vec{w}_2 \rangle$ such that for some $x \in L$ $\vec{w}_1 \cdot x \cdot \vec{w}_2 \in L$. A string $\vec{v}$ is a* ***constituent*** *in the context $C = \langle \vec{w}_1, \vec{w}_2 \rangle$ if C is basic and $\vec{v}$ can occur in C.*

Let us give an example. Consider the string

(2.31) *Peter gives a book to his friend.*

Substituting *Mary* for *his friend* yields a grammatical sentence, likewise substituting *Fido* for *a book*. We then have

(2.32) *Peter gives Fido to Mary.*

It is not possible to substitute anything for *to Mary* but we can replace *gives Fido* by *talks*. Still it is impossible to replace anything simple for *to Mary*. Finally, we can replace *talks to Mary* by *walks*. This results in the following analysis.

(2.33) *(Peter ((gives (a book)) (to (his friend))))*

This is actually the standard constituent analysis for this sentence. It is not always easy to verify that nothing can be substituted for a given string, but the analysis as put forward here yields an analysis in finitely many steps if the vocabulary is finite. Let us notice that there are strings that can appear both as a constituent and as a non-constituent, depending to the context. For example, *William and the firemen* is a constituent in (2.34) but not in (2.35).

(2.34) *William and the firemen talked to the president.*
(2.35) *Superman attacked William and the firemen helped him.*

A language is called *transparent* if no string can occur as a constituent is some context and as a nonconstituent in another context. Natural languages are thus nontransparent.

Now what about the classification of strings into categories? As we have seen, strings are not exactly the same as lexemes, because the latter are constituents in all contexts, while strings may show both behaviour depending on the context. Nevertheless, we wish to say that if a string is a constituent in a given context then it has the same type than the lexeme that can be substituted for the string. Behind that is the assumption that if it so happens that the same string appears in another basic context, the same lexeme will be substitutable for the string again. If we want to live without that assumption, the following definition has to be made.

Definition 2.3.3 *Let L be a language over the vocabulary V. The* ***syntactic type*** *or* ***category*** *of a string $\vec{v}$ is the set of basic contexts of L in which it can occur.*

This is a very important definition and a guiding principle for syntactic classification. Notice that it has some peculiar consequence. Standardly, it has been assumed that there are so-called *lexical categories* and *phrasal categories*. Every lexical category, written X^0, has a corresponding phrasal category, denoted by *XP*. Here X is a variable for lexical categories. In a sentence, every lexical element of category X^0 is contained in a minimal string of phrasal category. This category must be the corresponding *XP*. This is part of X-bar-theory. The terminology is confusing, however, in that it suggests that there can be no element corresponding to *XP*. But this is not true. In fact, we are suggesting here that all phrasal categories should have a lexeme of that category. (This is not exactly a consequence of the definition but it is observed to be more or less true.) For example, there are nouns such as *book* and noun phrases such as *William*; there are verbs such as *gives* and there are verb-phrases such as *walk*.

It is possible to show that the definition above will not yield a constituent structure in all cases. For example, consider the sentence

(2.36) *William has bought himelf a new, red, fast, super car.*

Let us concentrate on the series *new, red, fast, super car*. We can replace *super car* by *car*, then *fast car* by *car* after that *red car* by *car* and finally *new car* by *car*. This gives the following structure

(2.37) *(new (red (fast (super car)))*

On the other hand we can also replace *red, fast* by *fast* showing the first to be a constituent. We thus have the unexpected result that there are overlapping constituents. To get around this problem we need the notion of an *adjunct*.

Definition 2.3.4 *Suppose $\vec{v} \cdot \vec{x}$ ($\vec{x} \cdot \vec{v}$) occurs in the context $C = \langle \vec{w}_1, \vec{w}_2 \rangle$. $\vec{v}$ is an* ***adjunct*** *of $\vec{x}$ if both $\vec{v} \cdot \vec{x}$ ($\vec{x} \cdot \vec{v}$) and $\vec{v}$ occur as in C and are of the same category. In the construction*

$\vec{v} \cdot \vec{x}$ ($\vec{x} \cdot \vec{v}$), $\vec{x}$ is called the **head**.

The idea then is that adjuncts, rather than being grouped among each other as in *new, red, fast* must always be grouped with what they are adjuncts of. In other words, we need to assume that an *adjunct is never of the same category as the head*. Otherwise the procedure will not produce an unambiguous bracketing.

Another test for constituent structure is *coordination*. The test works as follows. In a context C, assume that $\vec{v}_1$ can occur as a constituent. Assume that $\vec{v}_1 \cdot and \cdot \vec{v}_2$ occurs as well. Then the latter occurs in C as a constituent since it can be replaced by the lexeme which witnesses that $\vec{v}_1$ occurs as a constituent in C. It is then assumed that (1) $\vec{v}_1 \cdot and \cdot \vec{v}_2$ has the same category as $\vec{v}_1$ and that (2) $\vec{v}_1$ and $\vec{v}_2$ have the same category. This is the basic theory of coordination. Notice that – unless we take this test by definition to be showing us constituents and their category – its correctness gives us a theory about the syntactic behaviour of the word *and* and similar words. The line we want to take here is that the behaviour of *and* is a matter of empirical study, and not part of a definition. The reason is that it is hard to pinpoint exactly the behaviour of coordination. Typically, it is assumed that *and* can take any two elements of identical category and produce an element of identical category. This is false given our definitions. We can coordinate freely noun phrases with identical case regardless of whether they are singular or plural, and regardless of their gender and person.

(2.38) *William and the firemen*
(2.39) *me and you*
(2.40) *Cynthia and Jack*

Notice also that the result of coordination shows different category. If two singular noun phrases are coordinated, the result is plural. Thus, it turns out that there is no simple description of the effect of coordination. Therefore it has to be used with care.

2.4 Categorial Grammar and Basic Phrase Structure

Categorial grammar has a long tradition and builds on the notion of *selection*. Selection is the term used for lexical elements which need other elements in order to form the next higher constituent. Since selection goes hand in hand with the building of the structure it should be no surprise that it also functions in the semantics of the selecting items. This is the basic insight of MONTAGUE. Originally, it has been assumed that the syntactic selection is just a reflex of the semantics and so ultimately syntax can be reduced to semantics. Although semantics can certainly be built up this way this makes little sense as a research strategy because there is more often than not no semantic reason for the observed syntactic behaviour. The semantics that one would have to postulate stand no way near what would intuitively be correct. This casts doubt on the assumption that syntax is a reflex of semantics.

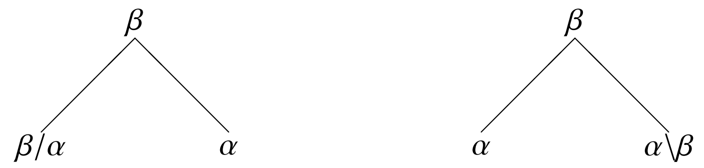
Definition 2.4.1 *A lexical element v **selects** an element of category $\mathfrak{C}$ to its **right** if in every context $C = \langle \vec{w}_1, \vec{w}_2 \rangle$ in which v occurs, there is a $\vec{x}$ and a $\vec{y}$ such that $\vec{v} \cdot \vec{x}$ is a constituent and $\vec{w}_2 = \vec{x} \cdot \vec{y}$ and $\vec{x}$ occurs as a constituent of category $\mathfrak{C}$ in $\langle \vec{w}_1 \cdot \vec{v}, \vec{w}_2 \rangle$. Analogously, **selection to the left** is defined. In the constituent $v \cdot \vec{x}$ v is called the **head** and $\vec{x}$ the **complement**.*

Compare the notion of complement with that of an adjunct. By definition, an adjunct is not selected. Otherwise it has no choice but to appear where it is. On the other hand, we will assume that the adjunct itself selects the head. This is true for adjectives as noun modifiers. In the position that they occur in they cannot be without the modified noun. In sum, when there is a constituent, we always find that there is one daughter which gives the reason for the other daughter(s) to be there and without which this daughter would not form a constituent at all. It is not necessary to assume that all constituent structure is binary branching, but our notion of selection would force us to assume that. It has been argued by some linguists that for example the verb *give* does not select first an indirect object and then a direct object, but that it does select both of them at once. It turns out, though, that the binary hpyothesis is by and large tenable. In categorial grammar this is enshrined in the notion of a category. [7]

Definition 2.4.2 *Let C be a set. The set $\mathfrak{Cat}(C)$ of **directional categories** over C is the smallest set containing C and which is closed under the formation rules*

($c/$) If $\alpha, \beta \in \mathfrak{Cat}(C)$ then $\alpha/\beta \in \mathfrak{Cat}(C)$.
($c\backslash$) If $\alpha, \beta \in \mathfrak{Cat}(C)$ then $\beta\backslash\alpha \in \mathfrak{Cat}(C)$.

The symbols / and \ are called *slashes*. The idea behind this definition is that any element which selects an element of category α to its right to form a category of type β is itself assigned the category α/β; and any element selecting an element of category α to its left to form an element of category β is assigned the category $\alpha\backslash\beta$. It follows that with the category of the argument and that of the mother given the category of the sister can be calculated uniquely. This gives the following trees.



(2.41)

We can express this as well with the equations

[7]Notice that we once more a different technical variant of the term *category*.

(2.42)	Forward Application	$\beta/\alpha \oplus \alpha$	$=$	β
	Backward Application	$\alpha \oplus \alpha\backslash\beta$	$=$	β

The nondirectional version of categorial grammar has only one slash, and we write $\alpha \multimap \beta$. Then we have the following rules

(2.43) $\alpha \multimap \beta \oplus \alpha = \alpha \oplus \alpha \multimap \beta = \beta$

Hand in hand with this goes the the assignment of meaning. If elements of the category are assigned a type $\mathrm{t}(\alpha)$ and elements of the category β a type $\mathrm{t}(\beta)$, then elements of the form $\alpha\backslash\beta$ and β/α are given the type $\langle \mathrm{t}(\alpha), \mathrm{t}(\beta)\rangle$. [8] To make this work, we need to start with basic types, for example *np* (to be of type *e*) and *s* (to be of type *t*) and assign bottom up categories to lexemes. Our first choice would be to put the category of intransitive verbs to be $np\backslash s$, that of transitive verbs $(np\backslash s)/np$, and so on. A transitive verb can then compose only with its direct object to form an intransitive verb, a sentence adverb like *necessarily* with a sentence to form a sentence etc. The basic calculus that we get is called AB, after Kazimierz Ajdukiewicz and Yehoshua Bar-Hillel.

However, as we have explained earlier, Montague assumed that the basic type *e* of objects is not the denotation of a noun phrase. His arguments were semantic in nature, but we can also give syntactic arguments for that. The observation to start with is that in some cases there is a mutual selection between elements. On the one hand an intransitive verb selects a noun phrase to its left, but by the same token a (nominative) noun phrase selects an intransitive verb to its right. This is the motivation behind the rule of *type raising*.

(2.44)	Right Type Raising	α	$\rhd$	$(\beta\backslash\alpha)/\beta$
	Right Type Raising	α	$\rhd$	$\beta\backslash(\alpha/\beta)$

The symbol $\rhd$ in the context $\alpha \rhd \gamma$ is read to be like an implication, to say that if an element has category α it has also category γ. Semantically, if x is a variable of type α and $\mathcal{P}$ a variable of type $\langle\alpha,\beta\rangle$ then $\lambda\mathcal{P}.\mathcal{P}(x)$ is a function of type $\langle\langle\alpha,\beta\rangle,\beta\rangle$. Indeed, we have $\lambda\mathcal{P}.\mathcal{P}(x) \odot \lambda y.f(y) = f(x)$; so it is the same result as $\lambda y.f(y) \odot x$. This means that we have found a semantic rule corresponding to the syntactic rule of type raising. The rule of type raising makes sense only if we give up the assumption that words or constituents must have a unique type, but that they have an initial type assigned to them and can raise it when necessary. This has indeed been proposed. But what could be the reason for raising the type?

Well, observe that the proposal given for type assignments works well for English, but it runs into various problems in languages that have different word order. Consider German, which allows the following sentence to be grammatical.

(2.45)	...,	*daß*	*das Geld*	*der Räuber*	*stahl.*
	...,	*that*	*the money*	*the thief*	*stole.*

Here we have the order OSV, object followed by subject followed by verb. The verb

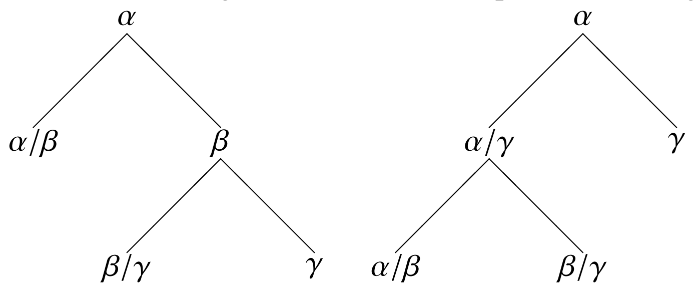
[8]Notice that there is no difference in meaning between two directional variants. We see exemplified the fact that there are syntactic facts that do not follow from any semantic type assignment.

may not combine with the subject because it would then assign it the status of an object, which is inadmissible for reasons of case. [9] What can be done in this circumstance is the following. We assume first of all that the following rules also hold.

(2.46)

Forward Composition	$\alpha/\beta \oplus \beta/\gamma$	$=$	α/γ
Backward Composition	$\alpha\backslash\beta \oplus \beta\backslash\gamma$	$=$	$\alpha\backslash\gamma$
Nondirectional Composition	$\alpha \multimap \beta \oplus \beta \multimap \gamma$	$=$	$\alpha \multimap \gamma$
	$\beta \multimap \gamma \oplus \alpha \multimap \beta$	$=$	$\alpha \multimap \gamma$

These rules have been motivated by Peter Geach. They are necessary in order to avoid overly complex categorization. For take the example of an adverb like *excessively*. It can modify both intransitive verbs, and also transitive verbs. If we say that *excessively* has category $(np\backslash s)/(np\backslash s)$, that is, are classified as adjuncts of intransitive verbs, they will by the above rule be able to act as adjuncts of transitive verbs. The corresponding semantical rule alias Frege-function is function composition. The way to see that this is correct we observe that if we were to feed the sequence $\alpha/\beta\oplus\beta/\gamma$ a γ to the right, we could compose β/γ and get β and then compose with α/β to get α. Instead the Geach-rule allows us to compose first α/β with β/γ to get α/γ and then compose with γ to get α.



(2.47)

Assume now that α/β has the meaning $\lambda x.f(x)$, β/γ the meaning $\lambda y.g(y)$. Then let u be an object of type γ. Then $\lambda y.g(y) \odot u = g(u)$ and $\lambda x.f(x) \odot g(u) = f(g(u))$. Now the composition $f \circ g$ of the functions is exactly defined by $f \circ g(u) := f(g(u))$. Thus, to associate with the Geach-rule the semantics of function composition is exactly as it should be from the viewpoint of categorial grammar. This hypothetical feeding of arguments plays a fundamental role in modern categorial grammar.

Now write V_0 for the category of sentences, V_1 for the category of intransitive verbs, V_2 for the category of transitive verbs, and V_3 for the category of ditransitive verbs. Then we have the following classification.

(2.48)

Subject	$V_1 \multimap V_0$
Direct Object	$V_2 \multimap V_1$
Indirect Object	$V_3 \multimap V_2$

[9]We are clearly using a semantical argument here. Per se, nothing permits us to ignore this in syntax. However, on syntactic grounds we can argue that the subject plus the verb do not form a constituent, because they cannot be replaced by a single item!

Regardless of whether V_0, V_1, V_2 and V_3 can be obtained by the basic set *np* and *s* we can observe that with the rule of function composition all six basic alignments in a sentence can be a (nondirectional) analysis which yields the category $V_0 = s$.

$$(2.49)\quad \begin{array}{lclclcl} \text{SVO} & : & V_1 \multimap V_0 \oplus (V_2 \oplus V_2 \multimap V_1) & = & V_1 \multimap V_0 \oplus V_1 & = & V_0 \\ \text{SOV} & : & V_1 \multimap V_0 \oplus (V_2 \multimap V_1 \oplus V_2) & = & V_1 \multimap V_0 \oplus V_1 & = & V_0 \\ \text{VSO} & : & V_2 \oplus (V_1 \multimap V_0 \oplus V_2 \multimap V_1) & = & V_2 \oplus (V_2 \multimap V_0) & = & V_0 \\ \text{VOS} & : & V_2 \oplus (V_2 \multimap V_1 \oplus V_1 \multimap V_0) & = & V_2 \oplus (V_2 \multimap V_0) & = & V_0 \\ \text{OSV} & : & (V_2 \multimap V_1 \oplus V_1 \multimap V_0) \oplus V_2 & = & V_2 \multimap V_0 \oplus V_2 & = & V_0 \\ \text{OVS} & : & (V_2 \multimap V_1 \oplus V_2) \oplus V_1 \multimap V_0 & = & V_1 \oplus V_1 \multimap V_0 & = & V_0 \end{array}$$

This is in fact as it should be. Notice that whether or not one must assume that we can impose restrictions concerning the order of the words, to account for English, for example, there are also languages which allow any of the six orders. For these the nondirectional version gives them a seemingly proper analysis. There are some quirks, however. In order to give an analyis at all we must assume that some pairs are constituents which are not constituents according to the definition assumed above. Moreover, we need to assume quite funny constituents such as pairs subject-object. Notice also that SOV has two analyses, (SO)V and S(OV), which corresponds to two constituent analyses. MARK STEEDMAN faces the problem head on in *Combinatory Categorial Grammar* (CCG) and declares that all of them are indeed constituents. His solution comes in combination with the assumption that in coordination we need such constituents if we want to analyse the following sentences.

(2.50) *Frank bakes a cake and proves theorems.*
(2.51) *Frank bakes a cake and Carsten rolls.*
(2.52) *Frank bakes and Imke eats a cake.*

If we take the analysis of coordination literally, that coordination is coordination of constituents then the three sentences give us all two element subsets of the string SVO as a constituent. Particularly worrying is that SO should also be constituent in the structure SVO. [10] We will later see how a solution can be given that avoids this paradox. For now let us note that the proposal still stands as a solution of the paradox. Let us see why it too cannot work.

Let us take a ditransitive verb like *give*. Let us work in a languages which allows for free word order among the four principal constituents. Latin is such a language. In this language, all 24 variations on the sentence (2.42) are grammatical.

	Carolus	*librum*	*uxori*	*dat.*
(2.53)	*Carl*	*the book*	*to his wife*	*gives*
	S	O	I	V

In particular the following serializations are grammatical.

[10] Combinatory Categorial Grammar also does not provide a constituent here; Steedman proposes a reanalysis of the SVO complex into two units, one of which is a consituent of same type as SO. This reanalysis is triggered by conjunction.

(2.54)	*Librum*	*dat*	*Carolus*	*uxori.*
	O	V	S	I
(2.55)	*Uxori*	*Carolus*	*dat*	*librum.*
	I	S	V	O

These are the only two serializations that are ruled out by the undirectional analysis together with the Geach rule. As before, some serializations receive several analyses, while some have only one, or in this case even none. The reason is that V can combine with I only, but S can combine with O, and O with I, but not I with S, S with V or O with V, and this rules out any analysis for the given string. Naturally, one has tried to overcome this weakness. Moreover, it is clear that this is only part of the problem. We still need to account for languages in which only part of the serializations are allowed. The methods to solve these problems are too involved to get into here. They show one weakness of the whole project, however. Categorial grammarians have been more or less lexically minded. They insist that grammar provides only very few and basic syntactical rules, while the power and richness of languages is the result of the lexicon in interaction with these rules. However, this only defers the problem to coding the behaviour of language into the lexicon given a sufficiently rich core of syntactical rules. It has been shown that what has become known as *Lambek Calculus* – named after Joachim Lambek – is equivalent in analytic power with context-free grammars, to which we will turn later. This result is due to Mati Pentus. The fact is, however, that once the behaviour of lexical items is known it is quite easy to deliver a context free grammar that accounts for this, so the question arises why we need categorial grammar for that purpose. The typical answer is that categorial grammar shows us a direct way to semantics. They point out that the categories one needs to derive the syntactical patterns can give us a direct clue as to how to set up the semantics, as has done Montague. Be this true or not, the important fact is that it gives us no clue as to what syntactical behaviour these elements have. Categorial grammar in its present day form has already lost its semantic innocence. Lexicalists experiment with complicated type constructors so as to get the syntactic facts right but they do not provide us any longer with such a principled insight that they have done earlier by pointing out that nothing permits to distinguish the combination transitive verb plus direct object from an intransitive verb. To the extent that this is false it teaches us that the truth has to be looked for elsewhere.

2.5 Morphology – The Additional Level

Let us pause and rethink the model of de Saussure in the light of the preceding discussion. We have isolated beyond semantics and phonology a third dimension of a sign, it *syntax*. Thus we write

(2.56) *give*

$/gɪv/$
$((np\backslash s)/np)/np$
$\lambda x\lambda y\lambda z.\mathbf{give}(z, y, x)$

The reason was that the semantic type does not give enough information about the proper combination of this item, in particular the directionality of selection. Thus, the reason for positing the additional level of syntax is that phonology has the directionality but lacks the tree structure to determine semantics via the Frege-principle; semantics has the tree structure but lacks directionality to get the phonology right. There have been arguments – for example in UCG but also within GB – that directionality has no locus in syntax but should only be present in the phonology. To make this work, the phonology of a sign cannot simply be assumed to be a string of phonemes, but a kind of template

(2.57) $p_3 + /gɪv/ + p_1 + p_2$.

Here p_1 is the variable for the phonology of the first argument, p_2 a variable for the phonology of the second argument, and p_3 a variable of the third argument. This correspondence must however be represented, and this causes the whole setup to change. We need to assume that what combines is the entire sign denoted here by *give*. We cannot separate a level of representation called *syntax* at which we may abstract away from semantical and phonological content and isolate all and only the combinatory properties of the elements. Syntax in our conception is the most general locus of combination which is expressed by the fact that syntactic structure allows two homomorphisms, one into phonology, turning the structure building operation into string concatenation, and one into semantics, turning the structure building operation into function application. If we are to follow the direction of UCG, these three levels get conflated into a single level of signs and combination of signs. Another way out, however, is to use two combination functions in phonology, forward concatenation with x and backward concatenation with x.

(2.58)

Forward Concatenation	$\lambda p.x + p$
Backward Concatenation	$\lambda p.p + x$

Thus, prepositions know syntactically only that they are looking for a noun phrase, but syntax is blind as to where the noun phrase will appear, to the left or to the right. This latter specification, however, is a matter of phonology. Although this is a way to go because it still recognizes three levels with their own mode of combination it does not conform to our conception of syntax. We will not dwell on this issue, however, simply noting that specifying the directionality in the phonology is a possibility.

A sign has three components, a *syntax*, a *semantics* and a *phonology*. This much is nowadays rather uncontroversial. It is the question, whether this is actually all there is to be said. We will analyse the situation by considering a difficult problem of linguistics, that of *analysis* or *segmentation*. It will eventually lead to a rather complex picture of language, one which linguists have over the years tried to reduce to the original model, with little success so far. The problem is caused by the fact that an additional level is assumed between the level of words and the level of phonemes, the level of *morphemes*. We will first outline arguments that lead to positing such a level and then proceed to an outline of a stratificationalist picture of language. Given that the mode of combination in phonology is concatenation it is plausible to say that the comparative of monosyllabic and disyllabic adjectives in English is formed by combining two phonological elements, namely the phonology of the adjective and the sequence /er/. Evidence may be the following list

	tall	:	taller	fast	:	faster
(2.59)	new	:	newer	long	:	longer
	short	:	shorter	high	:	higher

This can be read as a chain of oppositions. In each case the same semantic difference is paired with an identical phonological difference.

We therefore posit a decomposition of the comparatives into two smaller units. Since we cannot speak of /er/ as a word, for varying reasons, we are lead to say that it is an element of different character. Such elements are generally called *morphemes*. The lexemes *taller*, *smaller* etc. can and have to be segmented into two morphemes. One is the bare lexical entry, which we treat as unanalysable, and the other is the morpheme

(2.60)	*er*	/er/
		$(n/n)\backslash(n/n)$
		$\lambda\mathcal{P}.\mathbf{more}(\mathcal{P})$

n is the syntactic category of common nouns. Adjectives are premodifiers, so are of the category n/n, and *er* is thus of category $(n/n)\backslash(n/n)$. This is of course only a very rough approximation. In the first instance, a comparative adjective allows for a phrase opened by *than* as in *His car is faster than mine.* This is unaccounted for in the syntax as well as in the semantic, where **more** is just a placeholder for a semantic analysis that we do not intend to provide here. The standard definition of a morpheme is that *a morpheme is a smallest meaningful unit*. This definition is quite imprecise, however. It is unclear whether *smallest* refers to the phonological content or the semantic content. Since we will analyse the segementation problem below let us briefly have a look at the semantic segmentation. It is painfully clear that we cannot define a smallest unit of meaning in an abstract sense. How many units differentiate a table from a chair? Is the difference between a book and a leaflet minimal or not? Indeed, the impossibility of such problems have led structuralists to assume that a contrast is minimal if it is *within the prefabricated distinctions that the system makes*. But even that is plainly imprecise, because any natural language makes infinitely many distinctions, and any distinction can be broken down into a chain of several distinctions. Thus we are led to assume that if the definition of a morpheme is to make any sense at all it must involve the phonological content.

There are several facts that that get in the way of an analysis of the comparative into a sequence of two morphemes. First, morphemes such as */er/* demand a strict adjacency to the word to which they attach. There can be no element intervening between the adjective and the comparative morpheme. This can be captured by positing a difference between word boundaries and morpheme boundaries. We will say that the phonology of */er/* is / † ♯/ where † is a morpheme boundary and ♯ a word boundary. Then, as */er/* selects an adjective to the left it will attach only leaving a morpheme boundary while to the right it will induce a word boundary which permits no further attachment of morphemes. The additional morpheme boundary marker will have a significance that will become clear in later chapters. The second problem we have is that the combination rules are not as simple. We have, for example, that the result of combining */er/* with */simple/* is */simpler/*, which is pronounced without the epenthetic vowel called *schwa* between /*p*/ and /*l*/. The fact that this is so casts first doubts on the claim that phonology simply chains phonemes. In fact, it has been proposed that at the level of phonology there are rules which make combinations of phonemes better pronounceable, so-called *euphonic rules* or *Sandhi*. We might argue whether phonologically speaking we should posit such an epenthetic *schwa* at all. This will eliminate the problem in phonology but effectively defers it one level lower. Phonology will insist that /simple/ is pronounced as a sequence of the phonemes /*s*/, /*ı*/, /*m*/, /*p*/ and /*l*/. However, the phonetic realization allows for the insertion of *schwa* as a kind of grease to make the word pronounceable. But it can be shown that this cannot be a question of phonetics alone. The reason is roughly speaking that phonetics should be a universal property of humans; there should be no difference in Finnish phonetics from that of English. Nevertheless, there may be different phonology, because the phonology is an arbitrary abstract system. This is the reason why we will say that vowel harmony in Finnish cannot be a fact of phonetics. Perhaps there are properties of Finnish vowel harmony that are phonetically predictable, the particular way it operates cannot be said to be due to phonetics. One fact that is that German has (roughly) all the sounds that Finnish has without exhibiting vowel harmony. Consider then the case endings of Finnish. The inessive of *talo*, house, is *talossa*, meaning *in the house*. Likewise we have *ravintolassa*, in the restaurant, *bussissa*, in the bus. However, we have *hississä*, in the lift and others. The alternation between *ssa* and *ssä* is completely predictable from phonological laws of Finnish. Comparable argumentation concerning devoicing in German lead to the conclusion that the alternation between /*ya:g*/ and /*ya:k*/, meaning 'to chase', is phonologically determined. The first is used when the ending begins with a vowel (as in *wir jagen*, we chase) and the second when the ending begins with a consonant (as in *er jagt*, he chases). If we take the phonologically conditioned alternations into account, the segmentation into morphemes in the particular case of *er* is actually quite successful.

There are two more problems that the segmentation analysis faces. The hardest problem for the analysis so far is what has been termed *portmanteau realization* by Hockett. The adjective *bad* when combined with *er* yields *worse* and not *bader*. The other problem is that of *suppletion* or *allomorphy*. The adjective *good* yields *better* and not *gooder*. In the last case we can still segment into the sequence *bett* and *er*. That is to say, we will argue that this is form is still open to the same segmentation but we have to clarify the relation between *good* and *bett*. [11] The answer consists in positing a one-to-many relation of realization. We speak of a single morpheme, but say that it has possibly different *morphs* that realize it. Morphs are still abstract, but they are still phonetically visible. This means, in the case at hand we say that *bett* is an *allomorph* of *good*. In fact, we also have the possibility to say that it is an *allolex* since /good/ is the phonology both of the morpheme *good* and the lexeme *good*. This poses some problem for the identification that there are items which are both lexemes and morphemes inasmuch as their phonological realization is concerned. If the model is to make any sense at all, the two cannot be independent. Nevertheless, there is a question of whether we have identified an item qua lexeme or qua morpheme. This refers us to the problem of allomorphy. Can we decide whether we have to say that *good* is an allomorph of *bett* rather than an allolex? What criteria can there be to make such a distinction? One criterion is to analyse the cause of the alternation between the two forms. Analysing the case at hand we see that it is morphologically conditioned and not lexically. This is so because there is no syntactic process on the level of the usual lexicon that will induce a change from *good* to *bett-*. Rather it is only the combination at the morphological level that triggers a change from *good* to *bett*. We say that it is a *morpholexical* alternation. If that is so, we say that we are faced with a single lexeme *good*. However, being a morpheme, too, it displays phonetically different forms and this in turn means that the morpheme *good* has different allomorphs. There is a clear rule that determines which allomorph has to be taken in a given context.

This leaves the case of *bad* and *worse*. Here there is no plausible phonological or phonetic rule that allows a segmentation into two different morphemes. Thus, from a morphological point of view, *worse* must be a unit. On the other hand, there is no indication why we should not have the combination *bad+er*. Let us assume that we do have it, at least abstractly. The picture then is this. We have the lexeme *worse* which corresponds in meaning with the combination *bad+er*. The process that on the morphological level will give the analysis *bad+er* is then blocked by the fact that there already is a single lexeme, or for that matter also morpheme, namely *worse*. In that case, what we get as a phonetic realization of the morphemic sequence *bad+er* is that of *worse*. The latter is called a *portmanteau realization* of the morpheme combination *bad+er*.

The model that we have developed is that of *stratificationalism*, developed by SYDNEY LAMB. It is one of the most elaborate systems of structuralism and bears resemblance to *tagmemics*, developed principally by KENNETH PIKE. LAMB assumes that language operates on different *strata*. These are more or less abstract levels of description. We have at the moment a division into four levels, *phonology*, *morphology*, *syntax* and *semantics*. [12]

[11]Notice that the superlative is *best* which would leave us with a mere *b-* as a morpheme for the meaning of *good*. This may point to a solution via portmanteau realization, but the present line is nevertheless consistent. Crosslinguistically, as far as I am aware, the adjectives *good* and *bad* have three different stems in many languages, for the positive, the comparative and the superlative. Take for example Latin *bonus*, *melior*, *optimus*, and *malus*, *peior*, *pessimus*.

[12]Our outline is in several ways a simplification of the original model. However, the basic ideas can be made clear without introducing the full apparatus that stratificational grammar provides.

Each level supplies concrete elements and abstract elements. The terminology with which to refer to them is consistent. At the lowest level, phonology, we have the *phons*, which are concrete. The corresponding abstract elements are the *phonemes*. The relation between phons falling under the same phoneme is called *allophony*. The regime for putting phonemes together is called *phonotactics*. Phonemes do not bear meaning. One level up we find the *morphemes*. These are the abstract entities; their concrete counterparts are the *morphs* and the morphs falling under the same morpheme are called *allomorphs* of each other. Morphemes are combined, and the regime for doing this is called *morphotactics*. One level up we find the *lexemes*, *lexes* and *lexotactics*, mostly referred to as *syntax*. Still one level up we find the *sememes*, or units of meaning, and we consequently must find also *sems*, *allosemy* and *semotactics*. The levels are connected with each other via *realization*. A sememe is realized at the lexical stratum by some lexotactical complex. This in turn is realized on the morphological stratum as a morphotactical complex and the latter as a phonotactical complex. We assume that realization is defined only with respect to adjacent levels. Thus, a sememe cannot directly be realized as a string of phonemes or a string of morphemes. It must pass the lexical stratum first.

$$\begin{tikzcd}
\text{Semantic Stratum} \arrow[d, "\text{Realization}"] \\
\text{Lexical Stratum} \arrow[d, "\text{Realization}"] \\
\text{Morphological Stratum} \arrow[d, "\text{Realization}"] \\
\text{Phonological Stratum}
\end{tikzcd}$$

(2.61)

The division into strata might seem to be bewildering and counterproductive, but it constitutes an explification of what is the actual explanation linguists use in everyday practice. It leads to a great economy in explanation. Notice that the sounds corresponding to the sequence *ch* differ in German *Licht* and *Macht*. Since the change is determined by the context alone, there is no need to assume two different phonemes. This simplifies the representation in the mental representation of signs, since the distinction between the two need not be stored. Further, vowel harmony in Finnish has allowed us explain the two endings *ssa* and *ssä* as phonologically determined alternants of each other. Thus we are be exempt from positing two allomorphs. On the lexical level it is a bit harder to find similar cases. A very clear case is the use of classification nouns in Malay. We cannot say as in English *two clerks*. We must say *two men clerks*. The word *man* here serves no purpose but to identify the kind of object that is being counted. What distinguishes Malay from other languages that also have classifiers is that these classifiers are genuine words and have independent meaning in different syntactic environments. We can say *orang itu* *baca buku*, which means *This man reads a book*. In this sentence *orang* has its standard meaning *man*. However, when preceded by a numeral, it reduces to a classifier. Thus the numeral selects a classifier noun to its right. There is a fixed list of classifiers, which are determined largely by the noun that is being counted. Consider for example

(2.62)	*dua*	*orang*	*kerani*	
	two	CLASS	*klerk*	= *two clerks*
(2.63)	*dua*	*batang*	*rokok*	
	two	CLASS	*cigarette*	= *two cigarettes*
(2.64)	*dua*	*buah*	*buku*	
	two	CLASS	*book*	= *two books*

The analysis is to assume that there is a single lexeme best translated by *piece-of*, that intervenes between the numeral and the noun that is being counted. However, this lexeme comes in different forms. Crucial for our analysis is the fact that the selection of the form is determined by the shape of the object in question, so that the alternation must be said to be semantic in nature. Since there is no semantically driven morphological alternation (by lack of adjacency of the strata), the alternation occurs at the lexical level. Thus we have a case of allolexy. [13]

2.6 Underspecification and Sandhi

Earlier we have made allusions to the fact that phonology does not mirror morphotactical rules by mere concatenation, or to speak the language of mathematics, that the map + combining phonological strings cannot be simple concatenation. The problems are of varying complexity and have called for different types of solution. In this section we will look at the phenomena that we have called *Sandhi* or *rules of euphony*. These rules are phonotactical rules and therefore should get a genuinely phonotactical solution. Moreover, these rules are rules that operate when two items get in contact, so we assume that a proper formulation will involve only looking at the phonological junctures. To explain this, let us take a look at a string. It is a list of items, one following the other. If we take these items to be blocks that have to be glued together to form the string, then any time two blocks get glued we have a juncture. It is the moment of juncture when *Sandhi*-rules apply. However, if the blocks come already fully shaped there is no Sandhi without altering the shape of some of the blocks. This is roughly the view that is expressed by pro-

[13]The argumentation is not quite complete. For example, it is conceivable that the choice of classifier is not syntactically determined but rather semantical in nature, that is, what classifier is to be put depends on the shape of the objects. This is accounted for here by saying that the allolexy is semantically determined. Moreover, notice that the usual explanation of what classifier is to be put is in fact that one must look for the objects over which the noun phrase talks. Hence the stratificational grammar can only analyse this as a case of allolexy not allomorphy because the latter can only be lexically driven. This will then force us to assume that in languages where classification is morphologically marked we have lexically driven allomorphy.

ponents of a rules-and-derivations model, the most famous of which is the model of SPE (Sound Patterns of English). However, we can also think of the blocks alias phonemes as being underspecified before getting into contact and it is at the moment of juncture when the underspecification is removed because the context requires some feature to be this or the other way. This is the idea behind so-called *unification based approaches*. The latter view the syntactic entities (lexemes or morphemes for that matter) as specifying only some property of the phonological event that realizes them. Or, they specify exactly what is common to all phonological events realizing it. When elements get into contact they are a context for each other and the euphonic rules will determine some of the underspecified features.

We will motivate these ideas by considering two facts that we have met before, namely vowel harmony in Finnish and devoicing in German. The rule of vowel harmony of Finnish says that within a word, all vowels agree with respect to their *back*-feature. It is crucial for the proper statement of the facts that the word boundary is also the boundary of the vowel harmony. To be really exact, we need a proper definition of word boundary. For example, the following are genuine Finnish words.

(2.66) *henkilökohtainen*, *osakeyhtiö*

Since they are genuine Finnish words, they must analysed as compound words, with a boundary in between its parts. Although we can't hear this boundary as a pause, its effect is clearly detectable. We will write ♯ for this boundary even though it does perhaps not correspond to a word boundary in all detail. Derivational and inflectional affixes do not introduce a word boundary between the word they are affixed to, only a morpheme boundary. This will allow the spreading of the harmony from the word to the derivational and inflectional morphology affixes. We observe, namely, that it is the word that gets the affix that determines the harmony-feature, and not the affix. In syntactic terms, it is the stem that *governs* the affix with respect to the harmony feature. If we want to use a rules-and-derivations model here, we may actually assume that the affix comes with a default harmony, but that the latter is overridden by the governing stem. This is the intuition that we find in syntactic textbooks when they specify a *neutral form* for the affix, say *-ssa* for the inessive singular, and then go on to say that it changes when the stem displays *back*-harmony. On the other hand, if we use a unification based approach we assume that affixes are not specified for the harmony feature. Thus whenever they affix to the stem, the stem will determine the value of the harmony feature. But how does the stem get its harmony feature? The answer must in some sense be that stems are different from morphological affixes in that they must have a harmony feature while affixes may not. Whatever the reason for that property is, we still need to explain how the stem can know what harmony feature it has. This can be explained by underspecification theory only partly, and it is the latter part that causes some problems for the proper formulation of the vowel harmony of Finnish.

The guiding principle is that all that needs to be known is that stems determine, or *govern*, the harmony. They determine the shape of the affix. Each affix comes in two forms, one displaying *back*- and the other displaying *front*-harmony. Word and affix internally, however, there also is Sandhi, and we must account for that too. In other words, we must allow for the lexical specification to be such that lexical entries are not prefabricated, but allow harmony to operate on it. We might still concede, however, that there is a possibility for full specification, as for example in loan words, but that genuine words of the language are not fully specified. For normal words, however, we can say that it is not the individual phonemes that get marked for their *back*-feature, but it is the entire word. Since there are words which are compounds, this marking must be for each segment boundary-to-boundary. So we get something like the following representation.

$$\begin{array}{lccc}
(2.67) & \begin{array}{ccc} henkilO & \sharp & kOhtAIneN \\ - & \sharp & + \end{array} & \qquad & \begin{array}{ccc} OsAke & \sharp & UhtiO \\ + & \sharp & - \end{array}
\end{array}$$

Here, A, O and U are archiphonemes, underspecified for their *back*- or harmony-feature. Equivalently, they are the set of phonemes which agree in all but their back feature. For example, $A = \{a, ä\}$. Let us call A, O and U *strong* phonemes, all others *weak*. (There will e a distinctive feature *strong* to embrace that within the phonology.) Then the rule of harmony is that the strong phonemes govern the weak phonemes. This seems like an acceptable solution for the data. However, as is theoretically possible, there might be no strong phonemes within a lexeme. An example is the word *hissi*, in English *lift*. The correct form of the inessive of *hissi* is *hississä* and not *hississa*. Thus, even though there is no strong phoneme, or governor for harmony, nevertheless the harmony feature is unique; there is no choice as to how to assign it. This may be a reason for saying that there is a default for the harmony feature, which is *front*. Unless otherwise specified it assumes the value *front*. Alternatively, we can think of the following explanation. [14] We make the boundary markers ♯ and † into real phonemes. This is not implausible, because they too are part of the phonological system as we have demonstrated earlier on. Then say that the word boundary is strong and has the harmony feature *front* while the morpheme boundary is weak. Strong means now that it governs the value of the feature to the right. Unless a strong phoneme is present, the harmony feature of ♯ wins. However, the first, or in fact any strong phoneme wins over ♯. To make that work we need to assume that all phonemes can be marked for the harmony feature, although it will not always show up as a phonetic distinction. We say that we posit an *abstract* feature. This abstract feature can be passed

[14]I am inclined to call this a *trick* rather than an explanation. But arguments as the one I will present below are really the everyday method in linguistics and it is in principle hard to decide whether we have a genuine explanation or just found a trick to get the facts to follow. In the case at hand this can be decided on the basis of the question whether there are prefixes in Finnish. The formulation of Sandhi as proposed here can only work if stems are annotated in the lexicon with a boundary marker. This is turn implies that they must refuse either prefixes or suffixes. Since the standard affixes are suffixes, the proposal works as indicated. Notice also that we must distinguish two problems. One is the formulation of Sandhi, which is relatively easy. The other is to guarantee correct behaviour of affixes with respect to Sandhi in presence of the fact that stems are underspecified and need to recover the harmony feature from the phonotaxis. The present proposal is not meant as a definitive solution but as an illustration of how a solution can be obtained. If the data is more complex, a more complex solution must be found, too.

on along the phonological juncture '+'. Each word is divided into one or two parts. The first part is the half-open interval between the word boundary and the first strong vowel. The second part is the half open interval between the first strong vowel and the next word boundary. Call the first strong vowel in a word as well as each word boundary a *pivot*. Then the phonological string is divided as follows.

(2.68) $[\sharp\ldots]([\mathsf{p}\ldots])[\sharp\ldots]([\mathsf{p}\ldots])\ldots$

The brackets denote the *segments* determined by the pivots. The round brackets enclose optional segments. The rule of Sandhi is then as follows.

(2.69) *Each pivot determines the harmony in its segment.*

A word has two possibilities. It may consist of one segment or two. In the first case there is no string vowel present, and the boundary marker determines *front*-harmony. If there are two segments, the pivot vowel can determine the harmony in its segment. Since there are two kinds of pivot vowels, there are two kinds of harmonious strings, one with *front*-harmony, and the other with *back*. [15] The problem that we have is that a strong vowel needs to know whether it is a pivot or not. It is impossible to judge that on the basis of its environment, which consist (at most) of a phoneme to the left and a phoneme to the right. Again, an abstract feature must be introduced that allows for a phoneme to check whether it is eligible for being a pivot. The idea is that we have a toggle feature called *freeze*, which is passed left to right along the juncture. It is passed on negatively by $\sharp$ and positively by a strong vowel. All other phonemes may not change it. To define all these notions rigorously, let us introduce some terminology, which will also be helpful to understand syntactic processes. To formulate it we use booleans instead of features.

Definition 2.6.1 *A property P is said to* ***govern*** *a property Q* ***to the right*** *if in the combination $x + y$ of two phonemes where x is P, y must be Q. We write $P \curvearrowright Q$. P is said to* ***govern*** *Q* ***to the left*** *and write $Q \curvearrowleft P$ if whenever in $x + y$ y is P then x is Q.*

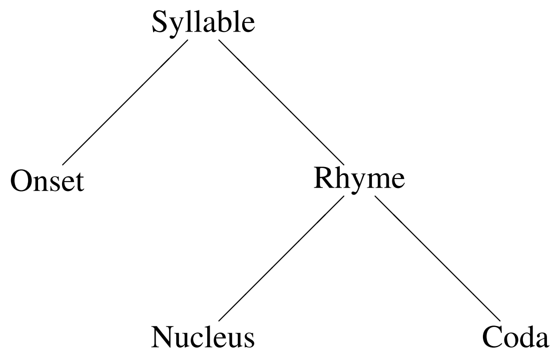
We need the booleans back, front, strong, pause, pivot and freeze. pause is true only of $\sharp$. front is the same as $\neg\mathsf{back}$. strong is true of pause and $/a/$, $/ä/$, $/o/$, $/ö/$, $/u/$ and $/y/$. pivot and freeze are contextually defined, via government.

[15]Notice that the formulation of Sandhi is not derivational in nature even though the metaphorical language suggest that it is.

$$\begin{array}{ll}
\textsc{Freeze} & \left\{\begin{array}{lcl} \neg\mathsf{freeze} \vee \mathsf{pause} & \curvearrowright & \neg\mathsf{freeze} \\ \mathsf{freeze} \vee \mathsf{strong} \wedge \neg\mathsf{pause} & \curvearrowright & \mathsf{freeze} \end{array}\right\} \\
 & \\
\textsc{Pivot} & \mathsf{pivot.} \leftrightarrow \mathsf{.pause} \vee \neg\mathsf{pause} \wedge \mathsf{strong} \wedge \neg\mathsf{freeze} \\
 & \\
\textsc{Sandhi} & \left\{\begin{array}{lcl} \mathsf{pivot} \wedge \mathsf{back} & \curvearrowright & \mathsf{back} \\ \mathsf{pivot} \wedge \mathsf{front} & \curvearrowright & \mathsf{front} \\ \neg\mathsf{pivot} \wedge \mathsf{back} & \curvearrowright & \mathsf{back} \\ \neg\mathsf{pivot} \wedge \mathsf{front} & \curvearrowright & \mathsf{front} \end{array}\right\}
\end{array} \tag{2.70}$$

The reader may check that the formulation does the job as intended. Notice that there might be more compact ways to write these rules down, by considering notions like *passing on a feature*. Such mechanisms will be considered in detail when we discuss GPSG, to which this solution owes a great deal.

Now consider finally the phenomenon of devoicing. In the same way as in Finnish we can assume this to be a process of spreading a feature. This time it is the *voiced* feature that gets spread. As before, it is not good to assume that lexical entries are simply underspecified at their end. For example, the naive approach to assume for roots of verbs such as *setzen* and *pfeifen* that they look like *seTZ* and *pfeiF*, where $T = \{t, d\}$, $F = \{f, v\}$ etc., so that the end is simply unmarked for the *voiced*-feature. This means that the value of the feature is not recoverable, so that for verbs like *jagen* and *toben* we will get *jaK* and *toP*, which would be identical for the verbs *jaken* and *topen*, if they existed. Thus we must assume that they all are marked for their *voiced* feature. However, the analysis can be enhanced by introducing a fine structure into the phonological string of syllables together with a subdivision of a syllable into *onset nucleus* and *coda*. The vowels are contained in the nucleus, and the consonants in the onset and the coda. A syllable can lack an onset, and it can lack a coda. [16]

(2.71)


We assume now that in German, the coda consists of voiceless consonants only. What explains the voiced/voiceless alternation is the fact that in different forms the final consonant cluster may or may not be contained in the coda.

[16]Unless, of course, we assume empty consonants.

(2.72)

j	a · g	e	n		j	a	g	t
O	N · O	N	C		O	N	C	C

It is clear that with this assignment of O, N and C the relevant facts follow. It is not clear whether we need to assume that the coda simply overwrites the value of the voiced feature, or whether we can formulate this in a unificational way. We will show that there is a unificational solution. Crucially, we add distinctive features for onset, nucleus and coda. It must be specified in what ways they can be distributed, and it is clear that lexical elements need largely not contain any information as to what the subdivision into syllables exactly is. However, the root of the verb *jagen* contain the following information concerning the final sound. First, it is either $/g/$ or $/k/$, that is, it is contained in the archiphoneme $/K/$. Moreover, if it is onset, then it is voiced.

(2.73) $[-\mathit{onset}] \sqcup [+\mathit{voiced}]$

Notice that the segmentation into syllables is at least as complex as vowel harmony in Finnish, and we have to omit that here. It is, however, in principle possible to spell it out in a similar way, using contextual features. In sum we have seen two things. One is that many facts have an explanation as soon as we assume a more articulate structure of phonotaxis, such as words and syllables etc. Second, there is a way to introduce abstract features into the phonotactic system that allows to state phonotactic restrictions as restrictions applying only at a juncture, that is, between adjacent phonemes. It is this latter observation that has led to the development of GPSG.

2.7 Discontinuity

In linguistics there exists also the notion of a *discontinuous constituent*. These are constituents which do not correspond to conxev subsets of the string. Given the previous definitions, the notion of a discontinuous constituent is a contradiction. However, to talk of such things is not to talk nonsense, because the problems that are raised in connection with discontinuity of constituents are real problems. We have met already one problem, that of *gapping*. Consider (2.44), repeated here as (2.74)

(2.74) *Frank bakes a cake and Carsten a roll.*

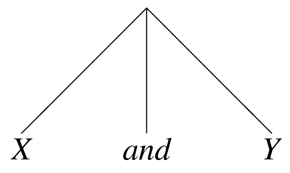
Under the plausible assumption that *and* only coordinates constituents we must assume that *Carsten rolls* is a constituent. Furthermore, under the likewise plausible assumption that *and* coordinates only constituents that are of same type (modulo the caveats that we have outlined above) we must assume that the corresponding constituent to the left is *Frank a cake* which is discontinuous. We can simply deny that the latter is a constituent, but the problem is to be solved somehow. The solution that has been proposed is to assume that there is a phonetically empty verb denoted by *v* that occupies the analogous position of *bakes* in the second conjunct.

(2.75) *Frank bakes a cake and Carsten v a roll.*

This solves our problem in the following way. Without such an empty element we have say why *Carsten rolls* is a constituent. If we assume that it is a sequence of two noun phrases then it is hard to see how they could possibly form a constituent. With this empty verb, however, we have a sequence SVO, and it is clear that it is a constituent. Notice that the argument is a mixture of syntactic and semantic arguments. For even if this empty element was present we still have the same visible object and have to call it a constituent. The constituency tests show that this is correct. Notice that the following sentences are grammatical.

(2.76) *Frank bakes a cake and leaves.*
(2.77) *Frank bakes a cake.*

This shows that the structure of coordination must be as follows.

(2.78)


The problem comes when we analyse the situation in conjunction with the semantic interpretation. Combinatory Categorial Grammar offers the solution of using function composition to get the interpretation of the combined noun phrases, but we have seen that it runs into problems with free word order. Positing an empty verb frees us from this problem, but gives us another: we have to make sure that it gets interpreted in the same way the visible verb in the left conjunct. So, from interpretive point of view we must get the same result as for the sentence

(2.79) *Frank bakes a cake and Carsten bakes a roll.*

Thus empty verbs allow us to uphold our notion of constituency and coordination while complicating the interpretation component by introducing a interpretive dependency between empty elements and visible (= overt) elements. This is a dilemma that we will quite often have; we get one thing straight at the expense of another. Empty elements are one such solution; this is why empty elements have not univocally been praised. Moreover, there has been harsh criticism against them. From a formal point of view there is no problem in assuming empty elements, and the only thing to show is that they allow in some way for a simple presentation of the ideas. We will have a lot to say about them in the subsequent chapters, and so we will seize the opportunity to present some alternatives.

A spectacular case of nonlinearity is the morphology of Arabic. In Arabic, roots commonly consist of a sequence of three consonants, out of which derived forms are produced by inserting vowels and manipulating some properties of the consonants. We will restrict ourselves to a simplified outline. Consider the root *ktb* which means *to write*. The following stems can be obtained from this root

(2.80)

katab	:	perfective active
kutib	:	perfective passive
aktub	:	imperfective active
uktab	:	imperfective passive

This list can be applied to all roots, in other words this phenomenon is regular. It is easy to see what the regularity is. And it is likewise easy to see that our previous approach to morphology will not do. The problem is that if we analyse these forms as sequences *root+affix* or *affix+root*, no matter which order we choose and no matter what phonology we choose, the realization cannot be of the form given above. The reason is that the root consists of three slots, while the affix consists of two vowels which have to be intercalated with the consonants of the root. Here is one description of the way in which the se forms are obtained. We can represent the root as *.k.t.b*, with three slots to be filled, and the affixes as *.a.a.*, *.u.i.*, *a..u.* and *u..a.*. The dots represent vowels in the root, and consonants in the affixes. In phonological terms we can say that the root is phonologically underspeficied at the dots, where it contains only the feature [+*vowel*] while in the affixes we find [−*vowel*]. It turns out now that both root and affix consist of slots, and the rule of combination is to put these slots into correspondence and take the conjunction of the information provided for each slot by the root and the affix. A strict condition is that the correspondence respects the ordering imposed on the slots by both templates.

(2.81)
$$\begin{array}{cccccc} \mathrm{V} & \mathrm{k} & \mathrm{V} & \mathrm{t} & \mathrm{V} & \mathrm{b} \\ & | & | & | & | & | \\ & \mathrm{C} & \mathrm{a} & \mathrm{C} & \mathrm{a} & \mathrm{C} \end{array} \qquad \begin{array}{cccccc} \mathrm{V} & \mathrm{k} & \mathrm{V} & \mathrm{t} & \mathrm{V} & \mathrm{b} \\ | & | & & | & | & | \\ \mathrm{u} & \mathrm{C} & & \mathrm{C} & \mathrm{i} & \mathrm{C} \end{array}$$

Notice that some slots cannot be made to correspond. The model can be made smooth by the introduction of an empty vowel, which we denote by ◦. Rather than *uktib* we would then have the form *uk◦tib*. We can then assume a regular six slot form for both root and affix. Moreover, if we insist on the syllable structure that is CV, consonant plus vowel, and an additional empty consonant, all morphemes have the same structure and matching consists in putting the templates over each other. Moreover, notice that the root gives information only over the nature of the consonants and the affixes only over the vowels. This has sparked off the idea that prosodic information is spread not over one linear order phoneme-by-phoneme, but over several such linear orders, called *tiers*. Each tier is a linear order with slots, and a tier contains specific information about specific distinctive features. There are now two modes of combination. One is concatenation, as before; the other is *unification*. In the latter case the two morphemes are interpreted as talking about the same phonological event. In the latter case the ideal situation is when the time slots that each morpheme provides are associable with each other one-by-one and that we can then interpret the associated slots as talking about the same phoeneme. Modulo the association of slots, the mode of combination is again unification.

In Arabic, we have been lead to assume that there is a vocalic tier and a consonantal tier. Tiers must be brought into correspondence; this means that we identify some slots in one tier with some slots in another tier. In general, identification across tiers $\langle T_1, \sqsubset_1 \rangle$ and

$\langle T_2, \sqsubset_2\rangle$ is a binary relation $R \subseteq T_1 \times T_2$ such that if xRu and xRv then $x \sqsubset_1 y$ iff $u \sqsubset_2 v$. This is called the *No Crossing Constraint*. It is a matter of direct verification that the simple model of consonantal and vocalic tier outlined above explains the data (2.56). It would be overly simplistic to declare that this is what there is to *Autosegmental Phonology*, since the data it sets out to explain is much more complex than (2.56). Nevertheless, it is enough for us to point out that it provides a model to generate data that would under the concatenative view of phonological realization be impossible to explain. We will have little to say about autosegmental approaches.

Chapter 3

Generative Grammars

We have seen how we can distill a structural description of a language by using a structuralist analysis. Now we turn to the question whether we can describe structure of the language in a succinct way. This will lead to the concept of a *generative grammar*. In its most elaborate form it is called *generalized phrase structure grammar*. We will see that many descriptions can be better described using movement transformations, which leads us to *transformational grammar* and to *head driven phrase structure grammar*.

3.1 Phrase Structure Grammars

Let us return to the definition of syntactic classes via context-sets. Suppose we are interested in a full description of all strings that can be generated within the language. That is, suppose we want to describe our language $L \subseteq V^*$. Such a description – in whatever terms – is what is traditionally called a *grammar*. A grammar is distinct from the *lexicon*. The difference is that the grammar is stable while the lexicon is an open set, that is, it can vary in time. From our structuralist point of view, the grammar does not deal with individual words but with word classes, and the lexicon only lists which words fall under which word class. Thus a lexicon is a list of pairs, consisting of an individual lexeme $v \in V$, and a word class, like $\langle /\mathrm{eat}/, V_2\rangle$, $\langle /\mathrm{walk}/, V_1\rangle$. We get at the word classes as follows. Recall that we have called v and w *syntactically equivalent* if their context sets are equal.

Definition 3.1.1 *Let L be a language over V. A* **word class** *is a set closed under syntactical equivalence. A word class is* **basic** *if it is the set of all words syntactically equivalent to a word in V. The* **lexicon** *of L, $\mathfrak{Lex}(L)$, is the collection of all basic word classes.*

It is clear that given the lexicon, we know which words are intersubstitutable in all contexts, and we can abstract from words and go over to word classes in the following way. We consider the map $\gamma_L : v \mapsto [v]_L = \{w | w \approx_L v\}$ assignig to each word its basic class. This map can be extended to a homomorphism of the string algebra $\langle V, \cdot \rangle$ by putting

$$(3.1) \qquad [\vec{x} \cdot \vec{y}]_L := [\vec{x}]_L \cdot [\vec{y}]_L$$

The claim is now that there exists a $M \subseteq [V]_L$ such that $L = \gamma_L^{-1}(M)$. This is not hard to verify. What we have to show is that $\approx_L$ is a congruence and that L is a collection of congruence classes.

Theorem 3.1.2 *Let $L \subseteq V^*$ be a language over V. Then $\approx_L$ is a congruence of the algebra $\langle V^*, \cdot \rangle$. Moreover, L is a union of congruence classes of $\approx_L$.*

Proof. We have $\vec{v} \approx_L \vec{w}$ iff $\vec{v}$ and $\vec{w}$ can be substitued for each other in all contexts. We show that $\vec{x} \cdot \vec{v} \approx_L \vec{x} \cdot \vec{w}$ as well as $\vec{v} \cdot \vec{x} \approx_L \vec{w} \cdot \vec{x}$ for all $\vec{x}$. Now, pick an $\vec{x} \in V^*$ and a context $C = \langle \vec{t}, \vec{u} \rangle$. Then put $C' = \langle \vec{t} \cdot \vec{x}, \vec{u} \rangle$ and $C'' = \langle \vec{t}, \vec{x} \cdot \vec{u} \rangle$. $\vec{x} \cdot \vec{v}$ can occur in C iff $\vec{v}$ can occur in C' iff $\vec{w}$ can occur in C' iff $\vec{x} \cdot \vec{w}$ can occur in C. Likewise, $\vec{v} \cdot \vec{x}$ can occur in C iff $\vec{v}$ can occur in C'' iff $\vec{w}$ can occur in C'' iff $\vec{w} \cdot \vec{x}$ can occur in C. Since C was arbitrarily chosen, the proof of the first claim is complete. Now let $\vec{w} \in [\vec{v}]_L$. Then if $\vec{v} \in L$, $\vec{v}$ can occur in the context $\langle \epsilon, \epsilon \rangle$. By assumption on $\vec{w}$ the latter can also occur in this context and we also have $\vec{w} \in L$. $\dashv$

This theorem provides a rigorous formulation of what is intuitively clear, namely that we can ignore the difference between words that are intersubstitable in all contexts. This has allowed us to separate the lexicon from the syntax. We will henceforth assume that a grammar is a description of $[L]_L$, that is of L reduced by the word classes. The big problem, however, is to provide such a description. Consider to that effect the set $\mathfrak{B}(L)$ of basic contexts. Recall that these are the contexts in which a lexeme can occur. We have used $\mathfrak{B}(L)$ to define the basic contexts to define the notion of a constituent. [1] Write $\mathfrak{B}_L(\vec{x})$ for $\mathfrak{C}_L(\vec{x}) \cap \mathfrak{B}(L)$. This set is the set of all contexts in which $\vec{x}$ occurs as a constituent. As before, we can define the notion of equivalence, but this time with respect to constituent occurrence. Write $\vec{x} \sim_L \vec{y}$ if $\mathfrak{B}_L(\vec{x}) = \mathfrak{B}_L(\vec{y})$. This abbreviates that $\vec{x}$ and $\vec{y}$ are intersubstitutable as constituents, not as arbitrary strings. To see the effect of this definition, consider the strings *William and the firemen* and *the firemen*. The two are intersubstitutable, but only if they occur as constituents.

(3.2) *William and the firemen talked to the president.*
(3.3) *The firemen talked to the president.*
(3.4) *Superman attacked William and the firemen rescued him.*

[1] One can disagree with the notion of a constituent that has been offered. Nevertheless, the approach that I present here is not dependent on the definition. One can define grammars simply by making reference to the constituency without knowing exactly how one arrives at it. It is our aim, however, to present a coherent system of definitions that yield concrete results.

(3.5) **Superman attacked the firemen rescued him.*

Thus, once again we see cases of nontransparency. The context sets defined by $\sim_L$ define congruence classes of constituents. This time, however, a reduction of the language, that is the strings modulo intersubstitutivity on the word level, cannot be compressed further. Instead, we can describe the language by saying how constituents are made up from simpler constituents. Call $\{\vec{w}\}_L = \{\vec{w} | \vec{w} \sim_L \vec{v}\}$ a *constituent class*. Each element of a constituent class is substitutable for another element of the same class if the first occurs as a constituent; the latter will then also occur as a constituent. Now if $\vec{w}$ is a constituent and can be segmented into subconstituents $\vec{x}_1 \cdot \vec{x}_2 \cdot \ldots \vec{x}_m$, then each subconstituent $\vec{x}_i$ can be replaced by any $\vec{y}_i \in \{\vec{x}_i\}_L$. Thus, the language is fully specified if we can list all possible ways in which a constituent can be segmented into subconstituents. Obviously, to do that it is engough to list all possible ways to segment a constituent into immediate subconstituents. Such a list is called a *phrase structure grammar* or *rewrite system*. We will concentrate in the remaining paragraphs on a special subcase, when we have only finitely many classes.

Definition 3.1.3 *A* ***context free grammar*** *is a quadruple* $G = \langle \Gamma, S, \Omega, R\rangle$, *such that* Γ *is a finite set, the set of* ***symbols***, $S \in \Gamma$ *the* ***start symbol***, $\Omega \subseteq \Gamma$ *the set of* ***terminal symbols*** *and* $R \subseteq (\Gamma - \Omega) \times \Gamma^*\rangle$ *a finite set, the set of* ***rules***.

Given G, there is a notion of *a labelled string generated by* G and that of a *labelled ordered tree generated by* G. In both cases the labels are from Γ. We take the second notion first. A rule ρ is commonly written not in the form $\langle A, B_1, \ldots, B_m\rangle$ but in the form $A \to B_1\ B_2 \ldots B_m$. That $\rho \in R$ says that a node with label A can for example immediately dominate m nodes, $y_1, y_2, ..., y_m$ such that $y_i \sqsubset y_j$ iff $i < j$ and the label of y_i is B_i. If that is so we say that the local tree headed by x instantiates ρ. Moreover, R is the exhaustive list of such possibilities. That is, a tree is generated by G only if every local tree instantiates one of the rules of G. We call the condition that each local tree must be instantiation of a rule of G the *local admissibility condition*. The local admissibility condition is only a necessary condition. Two more conditions must be added. The first is that the root is labelled with S (this is the *root condition*) and the the leaves are labelled with terminal symbols (this is the *leaf condition*). Notice that indeed the rules are statements of the immediate constituency. Now for the strings. We say that G generates the string S in zero steps. If it generates a string $\vec{x} \cdot A \cdot \vec{y}$ in n-steps and $A \to B_1\ B_2 \ldots B_m$ is a rule of G then G generates $\vec{x} \cdot B_1 \cdot B_2 \cdot \ldots \cdot B_m \cdot \vec{y}$ in $n+1$-steps. Then a string is *generated* by G if it can be derived in finitely many steps.

Definition 3.1.4 *Let* $G = \langle \Gamma, S, \Omega, R\rangle$ *be a context-free grammar.* $\mathfrak{L}(G)$ *is the set of strings* $\in \Omega^*$ *which are generated by* G. $\mathfrak{L}(G)$ *is called the* ***language*** *generated by* G. *A set* $L \subseteq \Omega^*$ *is called* ***(weakly) context free*** *if there exists a context-free grammar* G *such that* $\mathfrak{L} = L(G)$.

The relation between the trees generated by G and the strings is as follows. Given a labelled ordered tree $\mathfrak{T}$ we define a *string cut* or *section* to a maximal set of nodes such that no two nodes overlap. This section is called *terminal* if all nodes are leaves. There is only one terminal section, consisting of all leaves; it is often also called the *yield* of the tree. A section inherits a linear order from the tree, because two non-overlappping nodes are comparable via $\sqsubset$. Thus, we can view a section of an ordered tree as a string. Finally, if $\mathfrak{T}$ is labelled, the section also inherits the labels.

Proposition 3.1.5 *Let G be a grammar over Γ and $\mathfrak{T}$ be a labelled ordered tree with labels from Γ, σ a string over Γ. $\mathfrak{T}$ satisfies the local admissibility condition and the root condition of G iff all sections are generated by G. σ is generated by G iff there exists a labelled ordered tree satisfying the local admissibility condition and the root condition of G such that σ is a section of $\mathfrak{T}$.*

Proof. Let $\mathfrak{T}$ be a labelled ordered tree. Notice that there is a way to derive all sections by starting from r and replacing in a section $\vec{x} \cdot u \cdot \vec{y}$ u by the string $\vec{z}$ of its immediate daughters. If we look at the corresponding strings, we have that r corresponds to the string S (by the root condition), which is generated by G in zero steps. Now assume that $\vec{A} \cdot B \cdot \vec{C}$ corresponds to $\vec{x} \cdot u \cdot \vec{y}$. Assume that $\vec{A} \cdot B \cdot \vec{C}$ is generated by G. Then, by assumption on $\mathfrak{T}$, the local tree headed by u is an instance of the rule $\rho = B \to \vec{D}$ for some rule ρ of G. But then replacing B in the string by $\vec{D}$ yields a derivable string. Thus $\vec{A} \cdot \vec{D} \cdot \vec{C}$ is derivable. It corresponds to the section $\vec{x} \cdot \vec{z} \cdot \vec{y}$. Now assume that σ is generated by G. Then the tree of which σ is a section, will be defined by induction over the number of steps needed in the derivation. If this number is 0, then $\sigma = S$ and the tree is defined to the just the node r, with all relations empty. r is labelled S. Next assume that σ is derivable from $\tau = \vec{A} \cdot B \cdot \vec{C}$ in one step, by replacing B by $\vec{D}$. Assume that τ is derivable in n steps, so that σ is derivable in $n + 1$ steps. By induction hypothesis, there is a tree $\mathfrak{T}$ satisfying the admissibility condition and the root condition such that τ is a section of $\mathfrak{T}$. By throwing away all nodes properly dominated by a node of the section τ we can achieve it that we also have a tree whose terminal section is τ. It too satisfies the local admissibility condition and the root condition. For simplicity, assume that $\mathfrak{T}$ has all these properties and let $\vec{x} \cdot u \cdot \vec{y}$ correspond to $\tau = \vec{A} \cdot B \cdot \vec{C}$ in that decomposition. Then add to u a set of daughters y_1, y_2, ..., y_m and put them into correspondence with $\vec{D} = D_1 \cdot D_2 \cdot \ldots \cdot D_m$. Let $y_i \sqsubset y_j$ iff $i < j$. This defines $\mathfrak{U}$. $\mathfrak{U}$ satisfies the root condition and the local admissibility condition of G. Moreover, σ is a terminal section of $\mathfrak{U}$. $\dashv$

Trees define constituent structures, and so we can also speak of the *constituent structures generated by* G, which is nothing but the set of constituent structures of trees generated by G. Now, we want to say that a language L is strongly context free if there exists a context-free grammar generating the constituent structures of L. Notice, however, that this definition can be read in two ways. We can on the one hand say that the constituent structures are actually part of L, that is, we consider L as a set of constituent structures over V. On the other hand we can view the constituent structures as an something that we can derive from L by analysis. There are independent reasons to believe in constituents, but both readings are prima facie plausible. However, suppose that we have two different grammars generating the same language but with different analyses. Such is the case with the following two grammars, generating the language $a^+ \cdot b$.

$$(3.6) \qquad \begin{array}{l} S \to A\ S \\ S \to B \\ A \to a \\ B \to b \end{array} \qquad \begin{array}{l} S \to B \\ S \to T\ B \\ T \to A \\ T \to A\ T \\ A \to a \\ B \to b \end{array}$$

Then both represent possibly constituent analyses and should be admissible as syntactic analyses. This means that if we assume that constituent structures arise from substitution tests on strings that these tests yield no consistent answer, as we see above, a case corresponding exactly to what we have analysed in connection with adjuncts. There are several possibilities. We can simply choose one constituent analysis to be the official one and produce it via the grammar (as we have done with adjuncts). Or we say that all analyses into constituents are legal, in other words our string is *ambiguous* whenever there are two constituent analyses, and that the grammar must be capable of generating all these strings. This is the approach we will take here. Notice, however, that we also have the semantic interpretation as a criterion for constituent structure so that most cases of syntactic ambiguity can be resolved by looking at the semantic interpretation. To resolve the terminological clash, let us talk of a constituent structure $\mathfrak{S}$ as *projected by L*, $L \subseteq V^*$ a language, if it is a legitimate constituent structure from the viewpoint of substitution tests. A set of constituent structures $\mathfrak{C}$ is *projected by L* if it is the set of all constituent structures projected by L.

Definition 3.1.6 *Let V be a vocabulary and $\mathfrak{C}$ be a set of constituent structures. $\mathfrak{C}$ is called* ***context free*** *if there is a context free grammar such that $\mathfrak{C}$ is the set of all constituent structures of trees generated by G. Let L be a language over V. L is called* ***strongly context free*** *if the set of all constituent structures projected by L is context free.*

Notice that standardly one talks of a language L to be either weakly or strongly context-free. The latter concept make sense only if we assume L to consist not of strings but of constituent structures. Or, alternatively, we must say in what ways we can derive at a unique set of constituent structures that are defined by L. The above definition of projected structures achieves this.

3.2 Elements of Phrase Structure

Context-free grammars solve the puzzles of phrase structure that we have noted in connection with categorial grammar. The price to pay, however, is breaking the tight connection with the interpretation. However, to break free from the requirement to have a well-defined interpretation means to be able to approach first the constitution of phrases and worry later about how to square this with the semantics. This has been historically the line of thought. Therefore, let us temporarily ignore the problem of meaning and just attack the problem of phrase structure. Right from the start it seems plausible that the whole problem can be segmented into various subproblems. For example, we know that an intransitive verb only needs a subject to be a complete sentence. However, immediately we see that there is a complication. As the data below shows, there are two forms of *walk*, one for a singular, 3^{rd} person subject, and another for other types of subject. [2]

(3.7) *Fred walks.*
(3.8) **Fred walk.*
(3.9) **The children walks.*
(3.10) *The children walk.*

We can account for this in two ways. We can simply write two rules, one for singular, 3^{rd} subject, and another for other subjects. This will be rather tiresome, especially when we come to languages with rich inflection. In Latin, a verb shows six different agreement patterns, depending on number and person of the subject, some languages also have agreement in gender. For Latin, we would have to write down six different rules just to account for the present tense. This approach therefore does not get us very far and is rather unsatisfying, because it misses a generalization that we can descriptively capture by saying that subject and verb have to agree. Thus, we will say that there is only one rule, namely

(3.11) $V_0 \to NP \oplus V_1$

and that the complications are a kind of syntactic Sandhi, which is referred to as *agreement*. To factor out the effect of agreement from the rules of syntax we need to factor it out already at the level of categories. Analysing syntactic categories as attribute-value structures, we can posit a special feature called AGR which allows for features PERS, GEND and NUMB. The first takes the values *1, 2, 3*, the second *masc*, *fem*, *neut* and the third *sing* and *plu*. This is enough to account for Indo-European languages. The rule (3.11) can then be written with the help of a variable ranging over feature structures as follows

(3.12) $V_0 \to NP \sqcap [\textsc{agr} : \alpha] \oplus V_1 \sqcap [\textsc{agr} : \alpha]$

These rules have to be read in the following way. Any specific rule falling under this

[2]From the perspective of our approach we would not notice this as such, but see that there are categories *a*, *b*, *c*, *d* such that *a* can combine with *c*, not with *d*, and *b* can combine with *d* but not with *c*. The sketch here is outlined with hindsight, knowing that *a* and *b* are two forms of nouns, *c* and *d* two forms of intransitive verbs.

scheme for $\alpha = \top$ must not violate (3.12) for any more restricted value α can be assigned to one of the items. This interpretation has to be kept in mind especially when more complex cases are considered. Thus if we have a subject which is 3^{rd} person plural feminine, then α can be instantiated to that combination rather than $\top$, and the verb must then display the correct form under that value for α. Because of the limited range of values that α can assume, this will only account for the agreement facts, not more. Notice also that this rule is much more than just a rule of English; inasmuch as the facts about agreement are correct it accounts for all Indo-European languages and many more. The difference being that not all languages display the same type of distinctions, so that the range of values of α has to be adapted. In French, there is no neuter gender, just masculine and femine. In English, a verb shows no gender distinctions, but none of this fact casts doubt on the validity of (3.12). Notice that it requires that noun phrases have a *person*–feature instantiated although a noun phrase cannot choose which person it has. Nevertheless, a classification of noun phrases into first, second and third person exists, otherwise the agreement feature does not make sense.

We can continue in this vein; agreement between adjective and modified noun are accounted for by positing the following rule.

(3.13) $N \sqcap [\textsc{agr} : \alpha] \rightarrow Adj \sqcap [\textsc{agr} : \alpha] \oplus N \sqcap [\textsc{agr} : \alpha]$

However, two things must be noted. To account for the fact that a modified noun has the same agreement features than the noun itself, we must write $N \sqcap [\textsc{agr} : \alpha]$ both into the daughter and into the mother node. This is in contrast to the first rule where no agreement features have been annotated for V_0. Intuitively, this is because in the latter case it is the fact that agreement features have done their job, because they are features that mediate subject-verb agreement, so when the subject and verb are in contact, they can do that among themselves and the features are free at the mother node. In the case of adjective and noun we can better view them as features of cohesion, showing that this adjective modifies this noun and not another. We can also view as a mere consequence of the fact that adjectives are adjuncts. We will return to this point below. Notice that in the case of subject and verb agreeing we can view agreement features on a sentence, which appears here as V_0, is merely an artefact of the classificatory system. A sentence shows no agreement distinction, and that is all. [3] It can be demonstrated that sentences if they appear as subject in another sentence, are invariably 3^{rd} person singular. (3.14) and (3.15) gives evidence.

[3]The reader should also be alert to the fact that the discussion makes sense only if we abstract away from the terminology of *agreement*. If we talk about agreement, then it is a question of who agrees with whom. The fact that agreement features are primarily subject verb agreement and secondarily adjective-modified noun agreement features gives an indication as to how to set up the rules. Had we given the features a different name, then it would be not so clear as to how the rules should look like. The puzzled reader might picture himself exposed to a new language, where there is some systematic form of agreement. The question is whether he should call it *subject-verb agreement*. If so, the procedure for writing the rules is more or less clear. But the question is whether the analysis is correct. If not, the terminology is misleading, not the rules per se.

(3.14) *[To betray his friends like this]* **is/* are** *a shame.*
(3.15) *[That Fred betrays his friends like this]* **is/* are** *a shame.*

A simple sentence like (3.7) cannot appear in this position, however, so the evidence is not compelling.

However, we must also look at all the other constitution rules and see how they are set up. This is also a very time consuming procedure, and it seems that it too is amenable to a more succinct formulation. Recall in this connection the notion of an adjunct. X is an adjunct if whenever Y is of category C, then so is $X \oplus Y$. In other words, if we know that adjectives are adjuncts [4] then from the category of the sister we can predict the category of the mother (or conversely). In fact, an adjunction induces a strong form of agreement here, namely complete identity. All that needs to be specified is the category of the modified sister. Thus, we have two basic syntactic relations at work, *selection* and *agreement*. We can say as we did in phonology, that being an adjective governs to the right being a noun (in English). In semantical terms, we will say that the adjective *selects* a the noun. Since there are more types of adjuncts, verbal adjuncts alias adverbs, we can separate here a kind of rule skeleton involving only the notion of an *adjunct* from its specific instances where we have to talk about selection. The general rules is this.

(3.14) $\beta \to [\textsc{cent} : \mathit{adjunct}] \oplus \beta \qquad \beta \to \beta \oplus [\textsc{cent} : \mathit{adjunct}]$

β is a variable ranging over feature structures. Here, CENT is a new feature, called the *centricity* feature, which has four values, one of which is *adjunct*. As it stands, the rule is true by definition and adjunct, and therefore a universal fact of language, that is, it holds in no matter what language we pick. The only question is whether a word or constituent is an adjunct or not in the sense of the definition. Notice also that this pair of rules is equivalent with the definition of an adjunct (assuming binary branching), so that positing the rules (3.14) as universal rules is not nonsensical because there is no independent definition of an adjunct other than what is encoded in the rules. We can still abstract further from these two rules by writing down a single rule which does not state anything about the order of the daughters. The order between the daughters can then either be set within a language and universally for all categories or individualised to categories, or even individualized to words.

Now we turn to the centricity rules. To understand the intuition behind them it is necessary to unfold the nature of the syntactic nexus. Let us assume, as is standard in many syntactic theories but of marginal importance for the argument, that we have binary branching trees in syntax. Then, in contrast to phonology, the nexus involves three items, two daughters and a mother. Recall that the phonological nexus just involved two elements, namely two immediately adjacent phonemes. In syntax, however, matters are different because we have trees. Rather than a binary relation, the syntactic nexus is a ternary relation between nodes of a tree. In phrase structure rules this is directly encoded

[4] They are not necessarily. But let us assume that for the moment.

because it simply consists of a list of admissible local trees. And local trees constitute the syntactic nexus. In categorial grammar this is encoded by taking one daughter as the argument, the other as the function. Then the category of the function is $\alpha \multimap \beta$ which says that α is the category of its sister and β the category of its mother. If we allow items to carry a list of categorial types [5] then we can simply encode all structural relations that an item can engage in. In other words, the system of categorial grammar is only seemingly specific. Rather than classifiy the possible syntactic nexus qua ternary relation, it encodes the third argument, called *functor* via the category of the argument and the result. The functor carries the label *if-you-are-an-α-we-will-together-be-a-β*, nothing more. However, the notion of an adjunct is a different one. It says that the mother and the head are of same category. In categorial terms it will be of category $\alpha \multimap \alpha$ for certain α. But knowing that it is an adjunct and selects an α would be enough to know that it is of this type. Thus, the idea of a vertical encoding of categories has become more widespread. The idea is that in each nexus there is a mother *node* and two daughters, one of them the *head*. The head more or less determines the category of its mother. We say that the head *projects* its category. Projection is thus the vertical relation between head and mother, while selection is generally the horizontal relation. How projection works will be explained soon. If the non-head is an *adjunct* then the mother is of same category as the head. The case remains, however, when the non-head is not an adjunct. There is a case that we have already met, namely the case of a *complement*. A transitive verb takes two complements, a direct object and a subject. [6] A transitive verb plus a direct object is an intransitive verb. We have coded this before by numbers. We had a rough distinction between sentence (V_0), intransitive verb (V_1), transitive verb (V_2) and ditransitive verbs (V_3). There was a general rule of nexus. If the head daughter is V_{i+1} then the mother is V_i. This is, however, too rough because we have to specify also the category of complements that the verbs select, in particular their case. Thus, given the category V_0 as a primitive, calling it V, we can write V_1 and V_2 as

$$(3.15)\qquad V_1 := V \sqcap \begin{bmatrix} \textsc{select} : \begin{bmatrix} np \\ \textsc{case} : nom \end{bmatrix} \end{bmatrix}$$

$$(3.16)\qquad V_2 := V \sqcap \begin{bmatrix} \textsc{select} : \begin{bmatrix} np & & \\ \textsc{case} & : & acc \\ \textsc{select} & : & \begin{bmatrix} np \\ \textsc{case} : nom \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

In other words, we have used SELECT to write down in specific detail what complements the verb needs. We will then simply posit a rule that says that the mother is what the head

[5]In terms of categorial grammar, we say that the functor is polymorphous. Polymorphism is unavoidable in categorial grammar unless one opts for an underspecification approach and the use of variables. However, then the categorial approach gets close to undistinguishable from a phrase structure account such as GPSG and HPSG.

[6]The reader is asked to excuse me for calling the subject a complement of the verb. This is an artefact of the terminology set up here, which needs to be adjusted at certain places to accommodate for the finer distinctions. To the extent that the finite verb needs a subject, calling the latter a complement is justified.

daughter is minus the complement that the daughter selected. To encode this we need, finally, the notion of a *category*, as used in GB theory. This is a raw classification, abstracting away from morphological features, and complement selection. We will have to be content at the moment with a list; the *categories* are *noun*, *verb*, *adjective*, *preposition*, *complementizer*. More may follow. The category is the value of a feature CAT. Then the following rules are correct by virtue of the definitions.

$$(3.17)\qquad \begin{bmatrix} \text{CAT} & : & \alpha \\ \text{SELECT} & : & \beta \end{bmatrix} \quad \to \quad \gamma \oplus \begin{bmatrix} \text{CENT} & : & \textit{head} \\ \text{SELECT} & : & \gamma \sqcap [\text{SELECT} : \beta] \end{bmatrix}$$

$$(3.18)\qquad \begin{bmatrix} \text{CAT} & : & \alpha \\ \text{SELECT} & : & \beta \end{bmatrix} \quad \to \quad \begin{bmatrix} \text{CENT} & : & \textit{head} \\ \text{SELECT} & : & \gamma \sqcap [\text{SELECT} : \beta] \end{bmatrix} \oplus \gamma$$

However, in this formulation the rule is not quite correct. If we want to block the complement to pass on its own features, then it must not smuggle its selectional properties through the projection properties of the head. In the other hand, it is a fact that we must satisfy selection properties; thus we must assume the following additional axiom.

$$(3.19)\qquad [\text{CENT} : \textit{comp}] \sqsubseteq \neg[\text{SELECT} : \top]$$

The mechanism of this encoding is as follows. Each basic category is allowed to bear a list of complements, and it can combine with them in a fixed order. This is an advance over categorial grammar insofar as the rule schema is more general. However, free word order is still not accounted for. But let us continue for the moment with the development of abstract phrase structure until we pick up the matter again.

What we have achieved is an analysis of the syntactic categories into what we might characterize as the *bare categories*, plus morphologically marked distinctions, such as case, gender, number, tense, mood, and more; and finally, a list for subcategorization analogous to the SUBCAT-list used in HPSG. This allows for potentially infinitely many syntactic categories. Without the subcategorization list, however, there would be finitely many. It is assumed that a category cannot select for arbitrarily specific complements. In fact, what it cannot select is the selection properties of its complement, by the assumption that what it selects is the complement and we have just assumed that a complement when selected must have no unsaturated selection features. This is a strong restriction, but still does not restrict the possibilities to finitely many. But, as a quick inspection of a lexicon makes clear it is plausible that the subcategorization frame is not unrestricted. It must be. however, on the basis that we have a finite lexicon, because each word has a finite subcategorization frame. But this is not a satisfactory answer, because we want to know whether there are a priori limits to the length of such a list. In general it seems that the subcategorization lists do not exceed a certain length, which may either be thought of as given by the number of different morphological differences a language can make (with a restriction of two per type, so that two accusative objects would be possible, but not three) or restricted by the number of different primitive semantic arguments that can be specified (with a flat limit of one per type). The first is a restriction that derives for example from case marking properties, the latter from θ-marking properties, to use the terminology of

GB.

Finally, we come to the distinction between *lexical* categories and *phrasal* categories. This will be an opportunity to give an exposition of X-bar-syntax, by now a widely accepted skeleton of syntactic phrase structure. A category is *phrasal* if it takes no complements, only adjuncts. A category is *lexical* if it is a leaf which has taken only adjuncts. This might seem like an odd terminology, because it does not define these notions intrinsically, only with respect to a tree. But notice that the intrinsic categorization encodes part of the structure in which the element may occur. Given a leaf x in a tree which has basic category α, e. g. the noun *destruction*. It selects a complement. Therefore, it will always occur in a local tree with a complement, say

(3.20) *destruction of the city*

It can also take any number of adjuncts as in

(3.21) *violent, brutal, unprecedented destruction of the city*

Each of the constituents that contain the word *destruction* has the basic category *noun*. This is because these constituents form a chain such that each immediate subcontituent is the head of the next larger constituent. We say that these constituents form the *projection line* of the word *destruction*. We say that the entire constituent is the *phrase*, of which *destruction* is the *lexical head*, and that each projection between them is an *intermediate projection*. This has motivated to make a threefold distinction into *phrasal levels*, namely 0 for lexical, 1 for intermediate and 2 for phrasal level. From a categorial viewpoint, this cannot work as indicated, however. Roughly speaking, a verb with three arguments, say *give* must project four different levels, depending on the subcategorization frame.

(3.22)

gives	V_3
gives a book	V_2
gives a book to William	V_1
The president gives a book to William.	V_0

Standard X-bar-syntax has only the two ingredients, the basic category feature CAT and the level feature BAR. The first has the basic categories as values, the latter the levels 0,1,2. A category is *lexical* if it of level 0 and *phrasal* if it is of level 2. The universal rule skeleton is this.

$$(3.23)\quad \begin{bmatrix} \textsc{cat} & : & \alpha \\ \textsc{bar} & : & i+1 \end{bmatrix} \to \begin{bmatrix} \textsc{cat} & : & \alpha \\ \textsc{bar} & : & i \sqcup i+1 \end{bmatrix} \oplus \begin{bmatrix} \textsc{cat} & : & \beta \\ \textsc{bar} & : & 2 \end{bmatrix}$$

$$(3.23)\quad \begin{bmatrix} \textsc{cat} & : & \alpha \\ \textsc{bar} & : & i+1 \end{bmatrix} \to \begin{bmatrix} \textsc{cat} & : & \beta \\ \textsc{bar} & : & 2 \end{bmatrix} \oplus \begin{bmatrix} \textsc{cat} & : & \alpha \\ \textsc{bar} & : & i \sqcup i+1 \end{bmatrix}$$

Here, $i \in \{0, 1, 2\}$. Usually one writes $\overline{\alpha}$ for $[\textsc{cat} : \alpha] \sqcap [\textsc{bar} : 1]$ and $\overline{\overline{\alpha}}$ for $[\textsc{cat} : \alpha] \sqcap [\textsc{bar} : 2]$. (3.32) and (3.33) are the X-bar-skeleton. Commonly there are also unary rules, with the non-head omitted. However, we will assume that there are no such rules. This causes some trouble for the equation *lexical* = *level zero*, but gives more elegance to the system.

We assume that *Peter* is a noun phrase, and not of level zero. Likewise, intransitive verbs are to analysed as intermediate level verb projections, since they take only one argument, a subject. Some words can be either zero level or first level, e. g. words like *talk*, which have optional arguments. The centricity roles can be read off these rules with one important exception, namely that we assume $\beta \neq \alpha$ in the rule skeleton

$$(3.24) \qquad \overline{\overline{\alpha}} \quad \to \quad \overline{\overline{\beta}} \oplus \overline{\overline{\alpha}}, \qquad \overline{\overline{\alpha}} \quad \to \quad \overline{\overline{\beta}} \oplus \overline{\overline{\alpha}}$$

If we ban these rules, then centricity roles are fully characterized as relational properties within the X-bar-schemes. We have four roles, *head*, *complement*, *specifier* and *adjunct*. Every rules expands into a head and a nonhead. In other words, we assume the following basic scheme.

$$(3.25) \qquad \top \quad \to \quad [\textsc{cent} : \mathit{head}] \oplus [\textsc{cent} : \neg\mathit{head}]$$
$$(3.26) \qquad \top \quad \to \quad [\textsc{cent} : \neg\mathit{head}] \oplus [\textsc{cent} : \mathit{head}]$$

Finally, we say that the head projects its bar level iff the nonhead is an *adjunct*, that the nonhead is a *complement* iff it is not adjunct and its sister head is lexical, and that a nonhead is a it specifier if it is not an adjunct and the head sister is intermediate. To put that differently, the basic category of the mother is that of its head daughter, while the level has two possibilities. It can remain the same, in which case the nonhead is an adjunct, or it can go one up. In latter case it can go from 0 to 1, that is when we have a complement, or it can go from 1 to 2, and then we have a specifier. [7]

Although many people still adhere to X-bar-syntax, it is problematic in many respects and has been revised in one or the other form. In view of the evidence presented above it is not explainable why there should be a single specifier and a single complement. The arguments in favour of that are quite complex and will not be presented here but do not stand up to full scrutiny. Moreover, why should the schema (3.24) be banned? It seems more likely that the centricity roles are more fundamental than the bar levels and that the latter are recovered from the syntactic structure. The roles of complement and specifier are then properties that should be derived from the properties of the sister head. For example, we could argue that we two kinds of complements, the *specifier*, which is the complement of a head which subcategorizes for only one complement, so that the mother is a phrase, and the non-specifiers, which are all other complements. One reason for positing a specifier is that it takes part in agreement while the complement does not. A review of Indo-European languages shows that for subject-verb-agreement this is indeed so, taking the view that the subject is the outermost complement of the verb. Unfortunately, this does not account for adjective-noun agreement, and leaves out of consideration languages in which there is agreement between the verb and its subject

[7]Notice that we do admit zero-level adjuncts. Though this traditionally not viewed as part of X-bar-syntax, because the latter is in GB terms only a property of D-structure, such a line of argumentation is not open to us. If it should turn out that there is head-to-head movement that we can document by constituency tests (e. g. Dutch verb clusters) then the possibility of zero-level head adjunction is a surface fact and therefore part of the scheme, for we are taking surface structure as the basic criterion.

and it object. Basque is such a language.

The hardest problem for the universality of phrase-structure rules from X-bar-syntax is posed by coordination. It is the same problem that categorial grammar has with coordination. To account for the fact that the coordination compound is of identical category (more or less) than the individual elements left and right we must introduce variables into syntactic types or else admit massive polymorphism. Dito with X-bar-syntax. However, it is not unplausible to view coordination as a scheme on top of standard syntax, as a set of metagrammatical schemes that are valid throughout languages no matter what particular grammatical rules or rule schemes they have. The most common schemes are (3.27) and (3.28) with the coordinating element repeated throughout or showing up only once, at the end. The other examples (3.29) and (3.30) though not ungrammatical as such, have a special meaning, and can be counted as ungrammatical with the intended meaning. [8]

(3.27) *Peter, William, Theodor* **and** *the firemen*
(3.28) *Peter* **and** *William* **and** *Theodor* **and** *the firemen*
(3.29) *Peter* **and** *William, Theodor* **and** *the firemen*
(3.30) *Peter, William* **and** *Theodor* **and** *the firemen*

These facts can be acocunted for by very basic rules. We will not do that here since the facts at hand are too simple to allow for significant generalizations. Notice, however, that coordination points at a significant different between CASE and AGR. We see that the individual conjuncts must agree in case (this is not so clearly seen in English, but holds for example in Latin, Finnish and German) but they need not at all agree in their AGR-feature. However, there is nevertheless a regime for calculating the AGR-feature of the mother given the values for the daughters. This is a ternary relation between two daughters and a mother, and – as a rule – there is no possibility to reduce it to some binary relations. That means that coordination is principally of different type as the other nexus because the latter can be reduced to some binary relations, namely head-sister and head-mother modulo very basic rule schemata, the regime of centrictity roles, or else the basic X-bar-skeleton. Coordination is an intricate affair, to which we will devote our attention in a later chapter.

3.3 Movement, Part I: Adjunction

The rule skeleta of the previous section account for a number of syntactic phenomena, but they are still deficient. First of all, the problem of free word order is still not solved and secondly there are phenomena that can be quite frustrating for someone who tries to give a phrase structure account. ZELLIG HARRIS was the first to consider the idea that free

[8]Even under the intended interpretation they would not be counted as ungrammatical by a normal speaker, because the meanings of these alternative readings differ minimally, so that they will cause a minor violation at the interpretative level. Nothing hinges on the assumption of ungrammaticality, since this is only an illustration.

word order is not the effect of a phrase structure generation process but that the phrase structure rules define only one (or perhaps a small number) of the structures, and that the others are derived from this structure by means of what he called *transformation*. We will not make an attempt at retracing the original ideas but follow a path of thinking that has evolved during the last decades, due to Noam Chomsky and many others. In a language like German we observe that the order of the arguments a verb takes is free; the only restriction which applies is that the verb must be in final position, at least in subordinate clauses. What happens in main clauses will be considered below. An immediate solution that comes to mind is use the so-called ID/LP-format of rules, one which separates the part that determines the constituency from the that determines the linearization. We can simply say that a V_3 projects the following constituent

(3.34) $V_0 \to \{V_3, NP_1, NP_2, NP_3\}$

and state that verbal elements follow nouns. We know, however, that this does not generate the right constituents. We have given semantic arguments, but syntactic ones can be given as well. The boundary of a constituent can be found by considering where adjuncts can be placed. Since the constituents in question are verbal, consider the positions where adverbs can be placed.

(3.35)	*..., daß*	1	*Johannes*	2	*seiner Freundin*	3	*sein Auto*	4	*geliehen hat.*
	..., that		*John*		*to his girl-friend*		*his car*		*lent has.*

In fact, at all positions 1 – 4, and only those, adverbs can be inserted, and in any number, such as *nach langem Zögern* (*after long hesitation*), *gestern* (*yesterday*) and *ausnahmsweise* (*exceptionally*). Moreover, these adverbs can be spread over these positions. And this does not depend on the order of the arguments.

(3.36) *..., daß gestern Johannes seiner Freundin nach langem Zögern sein Auto ausnahmsweise geliehen hat.*
(3.37) *..., daß gestern seiner Freundin nach langem Zögern Johannes sein Auto ausnahmsweise geliehen hat.*
(3.38) *..., daß gestern sein Auto nach langem Zögern Johannes seiner Freundin ausnahmsweise geliehen hat.*

We have to conclude that rather than one quaternary rule we have several binary rules. If we want to account for this by general phrase structure rule schemes we can do this by positing not a list of arguments that the verb is looking for, but a *set*, and that each time it finds a complement, the corresponding item in the set will be removed. This will make sure that the verb can have its arguments in any order, one by one. Nevertheless, there are several things that can be noted. It is a fact that one of the serializations of the arguments is felt to be normal, while others are more and more marked, though not ungrammatical. Thus there is a default order of arguments. This can be argued to be a pragmatic effect depending on the nature of the arguments. But what should be the basis on which a normal word order is selected? It is not case, as is often assumed in the generative literature. (Haider, 1993) gives the following pair of verbs, which is a clear conterexample.

(3.37) *zumuten* NOM > DAT > ACC (engl. *to ask something of someone*)
(3.38) *aussetzen* NOM > ACC > DAT (engl. *expose someone to something*)

Rather, we see that there is a natural likeness from a semantic point of view of the argument that occupy analogous positions. The patient appears in dative case with *zumuten*, while it appears in the accusative case with *aussetzen*. Notice that English word order is morphologically determined.

There are arguments that show that the alternative word orders are not simply pragmatically marked but also let the original word shine through, as it were. Such arguments come from quantifier scope. Notice that normally, the quantifiers have scope over each other in the same way as they appear naturally in the string. Each quantifier takes scope over the next. This is the preferred reading. However, some sentences allow for a different reading, one in which the quantifiers do not take scope over each other the way their alignment dictates. Consider the following sentences.

(3.39) *Mindestens ein Stück von Stockhausen lobte jeder Kritiker.*
At least one piece by Stockhausen every critic praised.

(3.40) *Mindestens ein Stück von Stockhausen wurde von jedem Kritiker gelobt.*
At least one piece by Stockhausen has been praised by every critic.

(3.41) *Mindestens ein Kritiker lobte jedes Stück von Stockhausen.*
At least one critic praised every piece by Stockhausen.

(3.42) *Von mindestens einem Kritiker wurde jedes Stück von Stockhausen gelobt.*
By at least one critic every piece by Stockhausen has been praised.

In each of the sentences there is a universal quantifier in the scope of an existential, so we have a statement of the form $(\exists x)(\forall y)\phi(x,y)$. Thus we should be able to pick some x such that for all y holds $\phi(x,y)$. In the first two sentences we should be able to pick out a piece of music by Stockhausen such that it has been praised by every critic. In the second pair of sentences we must be able to pick out a critic who has praised all pieces by Stockhausen. This is the normal reading. Now some of these sentences allow for a different order of quantification, one in which rather than $(\exists x)(\forall y)\phi(x,y)$ we have $(\forall y)(\exists x)\phi(x,y)$ is true. To test this reading, consider that there has been a performance of *Sternklang*, *Inori* and *Alphabet für Liège*. Let there be three newspapers, in which the respective critic of the newspaper reports about this concert. Now let the critics be Jones, Smith, and Anderson. Suppose that Jones praised Inori but not the others, Smith praised Sternenklänge but could not find a good word for Inori, while finally Anderson adored Alphabet für Liège, while he disapproved of Sternenklänge. Then the question is whether we would still accept (3.39) or (3.40), even though under the normal reading they would be false. The contention is that we do accept (3.39), and that (3.40) we would accept with difficulty. [9] Now turn to (3.41) and (3.42). Under the normal reading they would be false, since no critic was full of praise for all the pieces, just some. On the other hand, under the reading with the order of the quantifiers reversed, they would be true. Here, the contention is that we would accept (3.42), but (3.41) only with difficulty. The explanation is in both cases the same. (3.40) and (3.41) are the natural serializations. (3.39) differs from (3.40) in that the object precedes the subject; (3.42) differs from (3.41) in that the *by*-phrase precedes the subject. If the different serializations are just some pragmatic variants, why is it that the quantifiers show some tendency to fall back into their original order?

What seems to be more likely is that there is only one set of phrase structure rules rule that generates the basic, default serialization and that the others are derived form that order by different means. Thus we may stick to the construction of the previous section, with the verb having a stack of selected items, which it may discharge one by one. This will inevitably generate a single word order, unless we assume the verb to be polymorphous. However, assume that there is a general process which is allowed to change the order of the arguments by moving them around. Then this would take care of the other word orders, and we would have an additional benefit of an explanation why the other word orders are marked; we could namely say that their markedness is the effect of a larger number of steps needed to generate them. However, as much as one would like to make such claims, they are very hard to demonstrate, and so we will use some 'harder' facts. One of them is the quantifier scopes; if is true that the quantifiers occupied the natural order in the course of derivation, then this could be a reference point for the marked readings other than the default ones. If we have a derived structure, then two quantifier scopes are possible, but if we have the the normal serialization we can get only one type of relative quantifier scopes, as the data above showed. So we have arguments for a two-layered process of sentences generation. First, we have a generation via phrase structure rules, and on top of that another process of rearrangement, called *transformational component*. The additional tranformational component opens a very promising view on cross-language variation. Unlike before, we can now assume that German *geben* and English *give* are of identical syntactical category. The different behaviour of the two can be explained by a conjuration of different factors. First, English has strict SVO order, while German has SOV order (in the subordinate clause) so that the basic serializations are different. [10] The second factor is the relative movability of complements. We can assume for example that German verbal complements are free to move leftward, while English

[9]These judgements also depend on the intonation contour one assumes for these sentences. The contention here is that with normal intonational contour, (3.39) allows for the alternate quantifier scopes, while (3.40) does not.

[10]Here an approach via an ID/LP-factorization of rules is plausible. We assume that the lexical entry of the verb specifies only the complement, not the order in which it stands with respect to this complement. In generative linguistics this has led to the claim that there is a parametric difference between language, depending on whether the verb selects its direct object to the right and whether it selects it to the left. Although this has fallen into disrepute, it still is a convenient tool in cross-language classification.

complements are subject to heavy restrictions. We will not spell out these matters in detail. Instead, let us review some of the facts noted above to get an idea of how movement actually works. The crux is that we observe a simple effect of movement, namely that it changes word order, but since our structures are not simply strings, we must also say what happens to the structure when we have movement. As we have seen with adverb placement, movement removes a constituent and creates another. For example, if we start from the basic structure SOV and move the object past the subject, we get OSV with the constituent structure (O(SV)). Originally, it has been assumed that movement means erasure of the constituent from where it had been and insertion into the new place. But a different view has turned out to square better with the theory. For we want the structure that has been obtained by movement to comply with the basic rule format of X-bar-theory. Since we have obtained X-bar-syntax on the basis of certain definitions plus evidence over all constructions of the language, derived or underived, we are compelled to assume that this is true. Thus, the structure (O(SV)) must conform to X-bar-syntax. But then we are in a dilemma. If it does conform to X-bar-syntax, then it is a V_0, that is, a sentence, and SV a V_1 lacking an object, and V_2 is a verb lacking a subject to form a constituent lacking an object. Thus the verb is polymorphous, after all. But this is exactly what the movement approach seeks to avoid. To circumvent this problem, we postulate an empty constituent at the point where the object had been before it moved. This is called a *trace*. The trace is phonetically empty, that is, inaudible, but it serves the syntactic purpose of giving the verb the constituent that it selects for. The trace frees the actual object from its location requirement by feeding itself to the verb. After that we project the verb phrase as usual. What remains to be stated are conditions that regulate the relations between the trace and the constituent it is related to.

(3.43) $[S\,[OV]] \rightsquigarrow [O[S\,[tV]]]$

Notice that there being traces or not, the visible or audible constituents are the same. Now we have to determine the category of the elements in question. By assumption, the trace must be of same category as the constituent it replaces. Hence, it is a noun phrase. It is selected by V_2 in the structure and the complex $[tV]$ is a V_1. It combines with the subject to form a V_0. Finally, we have to specify the category of the whole structure. Two things help. The first is the iterability of this movement.

(3.44) $[S\,[O[IV]]] \rightsquigarrow [I[S\,[O[tV]]]] \rightsquigarrow [O[I[S\,[t[tV]]]]$

The second is the fact that adverbs can be placed at all junctures, so that we must assume that the category of the additional constituents are verbal projections. Since we are at the phrasal level by the time the verb has found the subject, we must assume that the object and the indirect object in (3.40) are adjuncts of V_0. For by the fact that they are adjuncts, the category of the mother is that of the head, so we can iterate this process ad libitum. Second, the verbal character of the head is preserved, so that adverbs can be placed in between the moved constituent and the head. The movement type that we have described is known as *adjunction*, and the particular type of adjunction of arguments of the verb to the verb phrase as *scrambling*.

There is a lot to say about movement, and what there is to be said will fill large parts of this book. In particular, movement cannot just be the placement of a constituent somewhere else, otherwise phrase structure will be a total chaos. Restrictions have to be put. We will ignore that aspect of movement for now. Let us ask instead the question of what a trace is. By definition, it must be a constituent of the same category as the corresponding moved item. It is a kind of placeholder. We will assume, then, that languages can in principle have traces of any category. If a language fails to have traces of a certain category then there are elements which are not allowed to move – simply because nothing can fill the gap that they would leave. Moreover, we can ask about the structure of traces. Two possibilities are plausible. One is that they are just singular constituents without further structure. The other is to assume that they are structurally isomorphic copies of their antecedent. To decide between these options is very difficult and largely a question of elegance in the technical apparatus. For the moment we will adopt the view that they are singular constituents. The next problem we face is that of interpretation. If we assume Frege's principle, then traces can from an interpretive point of view not just be pieces of ink. After all, the verb selects a complement because it is a function that needs to apply to an argument. If we feed that argument in form of a trace, then the assumption of a uniform Frege-map makes it clear that we need to assume some meaning for a trace. In fact, we assume that traces are variables of the appropriate type. This also saves the type correspondence; we can still assume that V_2 corresponds to a function which will yield V_1 if applied to its argument, and that the latter yields V_0, a sentence, if applied to its argument. Although noun phrases are type raised, we assume here that their traces are not. [11] Hence, every trace of a noun phrase corresponds to a distinct variable, x_i, that we feed as an argument to the verb. Take for example the verb *den Hof machen*, *to court*. The following is an acceptable sentence with derived word order.

(3.45)	*..., daß*	*Sibylle*	*ein Junge von nebenan*	*den Hof macht.*
	..., that	*Sibylle*	*a boy next door*	*courts.*

We are led to conclude that the constituent *der Junge von nebenan den Hof macht* is a verb phrase, so it is not looking for a subject. The subject is present in form of a variable x_0, so that the translation is

(3.46) $\exists u.\mathbf{boy\text{-}next\text{-}door}(u) \wedge \mathbf{court}(u, x_0)$

Notice that this is an open formula, since the variable x_0 is free. This squares well with the intuition that *the boy next door courts* is not yet complete. But now we have to fit in the object. Originally, the translation of the object noun phrase *Sibylle* is

(3.47) $\lambda\mathcal{P}\lambda x.(\exists y)[\mathbf{sibylle}(y) \wedge (\forall z)(\mathbf{sibylle}(z) \rightarrow y = z) \wedge \mathcal{P}(y)(x)].$

The semantic type is $\langle\langle e, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle$, with $\mathcal{P}$ being a variable of type $\langle e, \langle e, t\rangle\rangle$ and x, y variables of type e. However, after extraction *Sibylle* is a verb-phrase adjunct; since verb

[11]This is a harmless assumption. If a trace is a variable, then it can also be viewed as function $\widehat{x} : f \mapsto f(x)$, as we have seen. So we might as well view traces as type raised, just like their corresponding antecedents. However, we will assume that type raising is syntactically triggered and that this trigger is absent with traces. But nothing hinges on that assumption.

phrases are sentences, the noun phrase must of semantic type $\langle t, t\rangle$, that is, it must be a sentence adverbial. To see how to get to this adverbial notice that all we need to do to obtain the intended meaning is to abstract again over the variable x_0 in (3.46). For if we attempt to plug in (3.46) as it is in the denotation of the object noun phrase we obtain a clash. Thus rather than inserting $\mathcal{P}$, a variable of type $\langle e, \langle e, t\rangle\rangle$, we insert an open formula containing the variable x_0. Moreover, notice that the λ-abstracted subject x has been fed already, so it does not have to appear here. Thus, finally, the interpretation of *Sibylle* at the moment of adjunction is

(3.48) $\lambda\mathcal{P}.(\exists y)[\mathbf{sibylle}(y) \wedge (\forall z)(\mathbf{sibylle}(z) \to z = y) \wedge (\lambda x_0.\mathcal{P}(x_0))(y)]$

Internally, we have abstracted over x_0. It is at this point that we need to know exactly which variable has been fed to the verb by the trace. Graphically, this is pictured by giving the trace an *index*, in this case the number 0, and giving the moved constituent, in this case *Sibylle* the same index, so that we know exactly over which variable to abstract. Now, if we feed to (3.48) the open sentence (3.46), we obtain the reading that we intend.

(3.49) $$\begin{aligned}(\exists y)[\mathbf{sibylle}(y) \;&\wedge(\forall z)(\mathbf{sibylle}(z) \to z = y)\\ &\wedge(\exists u)(\mathbf{boy\text{-}next\text{-}door}(u) \wedge \mathbf{court}(u, y)]\end{aligned}$$

This sounds satisfactory, but is only the beginning of a long chain of problems. The first is the fact that we have to propose numerous semantic types for one and the same noun phrase, depending on whether it is subject or object, whether it is in base position or has been moved. The second is the proper regime of the variables. The third is that we only derive the normal quantificational reading, and not the extra readings that are possible in case there are derived word orders.

The problem of semantic polymorphy is rather worrying for all proponents of categorial grammar. For it displays a fundamental waekness of the type raising approach. It may or may not be justified to posit a different semantic type of a direct object than for a subject, but we must also identify the trigger that helps to raise into these types if we want to uphold the uniformity of the Frege-map. Naturally, linguists have looked for such a trigger and have speculated that it might be (morphological or abstract) case. Consequently, we assume that a noun is segmented into stem and a case-morpheme, and the case morphemes have the following types.

(3.50) NOM $np \multimap ((np \multimap s) \multimap s)$
(3.51) ACC $np \multimap (np \multimap (np \multimap s). \multimap .np \multimap s)$

Unfortunately, we have seen that it is not possible to identify indirect or direct objects in German simply by means of case. We are led to conclude that the semantic type of a noun phrase depends on the type of the verb of which that noun phrase is a complement. If we want to avoid that, we are led to conclude that we had better given noun phrases a uniform type and let them initially be arguments of the verb. This is in fact possible; we can simply treat all overt noun phrases to be not in their original position, but in adjunct position to the verb. Then they are all of type $\langle t, t\rangle$, and their corresponding traces a are the arguments for the verb. This takes care of the variable handling, and also the type uniformity, but now we are back to square one. For now there is no way to distinguish positions that are derived from those which are not – all positions of overt noun phrases are derived.

The solution that all noun phrases are in derived positions, one which is assumed to be true for English in the *Minimalist Program* of Chomsky allows to encode syntactically two different notions; one is the association of arguments with variable slots in the semantic formula, which is managed through the discharge via function application in the building of the verb phrase. And the other is the view of a noun phrase as a sentence adverbial, which is the syntactic reflex of a *generalized quantifier* with the exception that a generalized quantifier is of type $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$, corresponding to a relation between sets. The way that a generalized quantifier yields a sentence adverbial is as follows. We have the generalized quantifier Q, and two properties, $\phi(x)$ and $\psi(x)$. Then $Q_x(\phi(x),\psi(x))$ is a sentence. Now, the denotation of a noun phrase carrying the index i is

(3.52) $$Q_x(\phi(x),(\lambda x_i.\psi)(x))$$

This gets the variable regime right; we abstract over x_i in ψ. Because ψ is a sentence and since it contains the trace t_i, the variable x_i is free, and we can meaningfully abstract over it.

The last problem is the hardest; assuming Frege's principle there is only one translation per syntactic structure. Thus, if one sentence allows different interpretations we must posit different syntactic structures with identical phonemic translation. To account for this, it has been argued that quantifiers do not anyway exhibit their relative scopes on surface structure and that they have to move even higher until their proper locus is reached. We will see later why this is reasonable. For now let us simply note that we can also interpret the different meanings as evidence of the fact that sometimes only the phonetic content moves, while the logical content (the one that interests the Frege-map) stays in situ.

3.4 Movement, Part II: Substitution

In this chapter we will review some classical facts about movement of so-called *wh-phrases*. The evidence for movement is more clear cut perhaps than in the case that we considered before, and is standardly used as prime evidence. It seems to me, however, that the basic word order is the most fundamental issue, and so I treated it first. Anyway, wh-phrases show different behaviour, and we will be led to conclude that the type of movement is different in that they do not move into adjunct position. Let us give the evidence.

English, otherwise a strict SVO language, requires the object to move at the beginning of the sentence if it is a wh-phrase. Thus we have both (3.53) and (3.54).

(3.53) *Who is visiting you?*
(3.54) *Whom are you visiting?*

Whereas (3.53) conforms to the usual sentence structure, (3.54) is out if the wh-phrase is replaced by a simple noun phrase such as *my friend* in a standard assertive sentence. If we assume the model of discharge of last section there is no question that the wh-phrase originates in the verb phrase and gets moved to the front. The question then is whether the result of this movement is the same as in the fronting of a direct object in German. This would be the case e. g. if we assume for English that it has wh-traces but not np-traces. If this is so then we conclude that if we have several noun phrases, all of them can be moved to the front of the sentence. But this is not so; look at the following data.

(3.55) *About what$_1$ are you talking to him t_1?*
(3.56) *To whom$_1$ are you talking t_1 about this?*
(3.57) **To whom$_1$ about what$_2$ are you talking t_1 t_2?*
(3.58) **Are you talking to whom about him?*
(3.59) **Are you talking to him about what?*
(3.60) **Are you talking to whom about what?*
(3.61) *You are talking to whom about him?*
(3.62) *You are talking to him about what?*
(3.63) *You are talking to whom about what?*

We see from (3.55) – (3.57), that of the wh-phrases only one can be fronted, and it must be if the verb precedes the subject ((3.58) to (3.60)), while it need not when the subject precedes the verb ((3.61) to (3.63), but only if used as an echo-question). It seems that with a three place verb we have not three, but four places to put arguments.

(3.64) *1 are 2 talking 3 4*

Assuming that the verb discharges its arguments in the canonical way, the positions 2, 3 and 4 have a natural explanation. But 1 is a new position, not projected by the verb. It is not an adjunct position to the verb, otherwise we assume that the process is iterable. Hence, we are led to conclude that the position is prefabricated by the sentence structure. Indeed, the verb *are* is a two place verb and so positing the structure in (3.65) solves the riddle.

(3.65) *1 are [2 talking 3 4]*

Now, the arguments originate as 2, 3 and 4. The subject (2) will move to 1 unless there is a wh-phrase, which will override this and move itself to 1. The details of this procedure go beyond the present interests. Moreover, we could have produced the same argument with the construction *do ... talk*, the normal way to produce questions from a sentence. The evidence points into the same direction; wh-phrases are not intrinsically forced to move, but one of them has to. The facts are the same in German.

(3.66) *Ich frage mich, wer dies gesehen hat.*
I wonder, who this has seen.

(3.67) **Ich frage mich, dies wer gesehen hat.*
I wonder, this who has seen.

(3.68) *Ich frage mich, was er gesehen hat.*
I wonder, what he has seen.

(3.69) **Ich frage mich, er was gesehen hat.*
I wonder, he what has seen.

(3.70) *Ich frage mich, wer was gesehen hat.*
I wonder, who what has seen.

(3.71) *Ich frage mich, was wer gesehen hat.*
I wonder, what who has seen.

In view of the fact that we have free order, this is strong evidence that in questions there must be a special position which can and must be occupied by a wh-phrase. The precise determination of that position requires long argumentation. However, we can show quite clearly that this position cannot be inside the verb phrase. For if so, adverbs would be allowed to precede the leftmost wh-phrase. But this is contrary to fact.

(3.72) **Ich frage mich, zuerst wer dies gesehen hat.*
I wonder, for the first time who this has seen.

(3.73) *Ich frage mich, wer zuerst dies gesehen hat.*
I wonder, who for the first time this has seen.

(3.74) *Ich sagte ihm, daß zuerst Einstein dies gesehen hatte.*
I told him, that was first Einstein to see this.

(3.75) *Ich sagte ihm, daß Einstein zuerst dies gesehen hatte.*
I told him, that Einstein was the first to see this.

To cut a long story short it is now a wide consensus that a sentence is not equal to a V_0, but that the V_0 is the complement of a new head different of a verb. This head projects a phrase with head, along with a complement and a specifier, and it is the specifier position that the fronted wh-phrase occupies. The nature of the head is difficult to determine and has caused some concern. It is now widely assumed correct that this head is a *complementizer*. The complementizer is normally not visible, but there are overt complementizers such as *that* and *whether*. Complementizers are what is called *functional elements*. Functional elements, or also called *closed class elements*, are of a quite restricted and invariable number and serve largely syntactical purposes. Nevertheless, they do have a meaning. A complementizer is the locus of illocutionary force. It determines whether the sentence is a statement, or a question or a promise etc. Thus, while the V_0 has the semantic type of a proposition (a thing which has a truth value), a sentence must have the semantic type of an *utterance*, which is roughly a proposition endowed with a force by the speaker.

With questions we do not have a clear semantic intuition as to what they mean, that is to say, logicians have not looked at questions as hard as they have been looking at assertions (or propositions, in fact) so that up to now we do not know well enough what they mean. Therefore, only a rough solution can be given here. First, let us take our initial examples.

(3.76) *Who is visiting you?*
(3.77) *Whom are you visiting?*

In each case the question is an implicit statement of the speakers ignorance about a certain state-of-affairs and a wish on behalf of the adressee to tell him what he knows about that state-of-affairs. The question is put by using an open proposition. [12] Let $\phi(x)$ be an open proposition. Then denote by $?x.\phi(x)$ the question *for which x does* $\phi(x)$ *hold?*. In the first example $\phi(x)$ is the open proposition *x is visiting you* and in the second example it is *you are visiting x*. Seen this way, the question operator ? functions just like a quantifier. Unlike *who*, *which* is a generalized quantifier rather than a quantifier. In its case the question operator ? must be a generalized quantifier acting in an analogous way. $?x.(\phi(x), \psi(x))$ will denote the question *which x which are* ϕ *are also* ψ*?*. The answer to such a question is a *list*, containing all those ϕers which are ψers, according to the addressee. But there are important differences. A question operator cannot be iterated in the same way as an ordinary quantifier. When it is, it means something totally different. The reason is roughly that we can have per sentence only one force. Consider

(3.78) *Who is visiting whom?*

The speaker is not aware of which people visit which people, and so he puts this as an open proposition. The appropriate answer is not a list of people who were visiting someone plus a list of people who have been visited, but a list of pairs of people $\langle x, y\rangle$ such that x is visiting y. From a logical point of view a more accurate translation of (3.78) is

(3.79) $?\langle x, y\rangle.\mathbf{visiting}(x, y)$

It is conceivable that there are ways to arrive at this translation by closer inspection of what the correct semantics of questions is, so that it would turn out to be equivalent to

(3.80) $(?x)(?y)\mathbf{visiting}(x, y)$

But this is only speculation for the moment. The fact that multiple wh-phrases give rise to a single question, asking for a list of tuples, and not separate lists, has led to the conception that wh-phrases all eventually move together into a single wh-constituent marking the objects that are being asked. For notice that wh-phrases do not display their scope at the surface and neither can their original, underived position tell us about it. Rather, they all have a common scope, the main clause – unless we are talking about embedded questions. To account for that, something similar to the in situ interpretation of quantified phrases has been invented. It is assumed that the reason a wh-phrase can take such a wide scope is that at one stage it has really been there. In the case of questions, however, this position

[12] Yes-No-questions are left out of consideration here.

must be leftward of the overt phrase. The problem is that the phrase has never been there before. Thus it was claimed that after being positioned for surface structure the phrase moves up from its surface position to that position where can take its natural scope. When it is there, the structure thus obtained is called the *logical form*, abbreviated LF. It should not be confused with the translation into logic that we are using here. The introduction of LF has led to a three-stage model of GB, also implicit in the *Minimalist Program*. The initial stage, called D-structure is the sentence structure conforming to the selectional restrictions, projected mainly by the latter. D-structure is generated by the context-free base component called X-bar syntax. After D-structure has been generated, the elements start moving around. At a certain moment the arrangement is spelled out, that is, the words have reached the relative positions that we can see or hear, that is when we have reached S-structure. The S-structure of the sentence is passed on to the phonological system for pronounciation. However, this is not the end of the movements. For now the words have to move to find their respective scopes. Only when that is achieved, the derivation stops, and we are at LF.

(3.81)
$$\begin{tikzcd}
\text{LF} & & \text{PF} \\
& \text{S-structure} \arrow[ul] \arrow[ur] & \\
& \text{D-structure} \arrow[u] &
\end{tikzcd}$$

3.5 Movement, Part III: Head Movement

The types of movement transformation that we have considered in the previous two sections have a lot in common. First, the constituent that is moved is an entire phrase, and second the landing site strictly c-commands the site of the trace. Here, a node x *strictly c-commands* a node y in a tree if neither dominates the other, and the mother of x dominates y. [13] The difference between them is that substitution is movement into a position that must be there for structural reasons, such as the specifier of the complementizer phrase while adjunction movement was into a position which does not have to be there and can be added in any number, i. e. into an adjunct position. The typology of movement operations is not complete. There is one very special type of movement that still needs to be added, namely *head movement*. A spectacular case of head movement is verb movement in Germanic languages. What we find (except in English) is that in the subordinate clause

[13] If x does not dominate y then x is not the root, and so there is a mother of x.

the verb must occupy the final position while in the main clause it appears as the second constituent.

(3.82) *Ich weiß, daß er den Präsidenten heute* ***trifft****.*
I know that he the president today will meet.

(3.83) *Heute* ***trifft*** *er den Präsidenten.*
Today he will meet the president.

Because of this one is initially inclined to classify German as a SVO language. Moreover, it is not a priori clear that the verb should originate at D-structure in final position. We could also say that it originates in second places and must move to the end in subordinate clauses. Finally, we could deny there to be any uniform D-structure position for the verb, and say that the verb is in second place in main clauses but at the end in subordinate clauses. Such must be the approach of non-transformational theories such as GPSG or HPSG. Even though this is in principle an unrefutable position, we will see that the facts point into a different direction.

First of all, note that it is not the verb phrase that moves into second place, but only the verb carrying finite inflection. Thus it is a different type of movement because what is moved is not a phrase. To argue for that it is not enough to show that we see only the verb move while its complements stay behind. We are already in the uncomfortable position that the complements may have moved themselves before the verb is moved, so that the verb phrase is already empty with the exception of the head, at least up to the specifier position. [14] We will present several arguments. First, consider the fact that (3.85) and (3.86) are not grammatical. This shows that the canonical position of the arguments at D-structure is not between *unterschrieben* and *hat*. Nevertheless, in the main clause it is only *hat* that moves, not the complex *unterschrieben hat*. But the latter selects the complements as a whole, so is a constituent at D-structure. However, at surface structure the two are separated. Since the previous types of movement do not allow to split up constituents, we must assume that the separation is an effect of movement by one of the constituents.

(3.84) *Es ist erwiesen, daß er den Vertrag* ***unterschrieben hat****.*

[14] Here, the distinction between inclusion and exclusion sticks out its head. Adjuncts still belong to the verb phrase in the sense that there is a projection of the verb that dominates the adjunct. On the other hand, the part dominated by the least phrasal node above the verb (the mother of the specifier of the verb phrase) contains the set of constituents that are more closely related to the verb than the adjuncts. We say the latter form the *strong constituent*, while the verb phrase in its entirety, with all adjuncts, forms the *weak constituent*. The problem here is that if only strong constituents can move then the fact that the complements may not be there, only their traces, may create the illusion that when the strong verb phrase moves it looks as if only the verb moves. Our problem is to refute the prima facie possibility that this is what is going on, and that it is really *only the head* that moves, and that it leaves behind a verb phrase whose strong constituent is now completely empty!

It is proved that he did sign the treaty.

(3.85) **Es ist erwiesen, daß er* ***unterschrieben*** *den Vertrag* ***hat****.*
It is proved that he signed the treaty has.

(3.86) **Es ist erwiesen, daß den Vertrag* ***unterschrieben*** *er* ***hat****.*
Is is proved that the treaty signed he has.

(3.87) *Er* ***hat*** *den Vertrag* ***unterschrieben****.*
He has the treaty signed.

(3.88) *Den Vertrag* ***hat*** *er* ***unterschrieben****.*
The treaty has he signed.

(3.89) **Den Vertrag er* ***hat unterschrieben****.*
The treaty he has signed.

(3.89) repeats the evidence that the verb is really the second constituent of the main clause. Now we have shown that it cannot be the previous types of movement that generate these structures, if we uphold the hypothesis that the different structures are generated via transformations. We have also seen that it is not the verb phrase that moves. Still, *unterschrieben* or *hat* may be analysed as phrases of some sort. Which of the two has moved? An answer can be obtained by looking at the places of adverbs.

(3.90) *1 Er 2 hat 3 ihr 4 sein Auto 5 geliehen 6.*

Only positions 3,4 and 5 allow placement of adverbs. Hence it is there that we must posit verbal constituents. If we assume the verbal head to be in second position we would have to assume that at position 2 we have a verbal consituent boundary. But adverbs may not be placed there. Adverbs are left adjuncts (in subordinate clauses), and so they turn out to be right adjuncts in the main clause. Thus, assuming the second position to be the underived position gets us into a series of unexpected complications.

Thus, we conclude that what we have is movement of the verb from the final position into second position. We will present two rather murky facts about head movement that support this conclusion. The first is that in main clauses verbs consisting of a preposition plus a stem like *weg+laufen* (to run away) or *auf+essen* (to eat up)[15] or a frozen complement plus verb like *rad+fahren* (to cycle) or *maschine+schreiben* (to type) split at the juncture.

(3.91) *Er* ***aß*** *seinen Apfel* ***auf****.*
He eat the apple up.

(3.92) **Er* ***aufaß*** *den Apfel.*
He eat up the apple.

(3.93) *Er* ***schrieb*** *den ganzen Tag* ***maschine****.* (colloquial)
He typed the whole day.

[15]But not *unter+schreiben*, lit. to ‘underwrite’. This verb does not split in this way; see below.

(3.94) **Er* ***maschineschrieb*** *den ganzen Tag.*
He typed the whole day.

The split is obligatory as the data shows. Moreover, the verb is at second place, and the preposition cannot be, likewise with the frozen complement. This is already an uncomfortable fact, because in order to explain it properly we have to produce a morphological analysis. On the other hand it shows that the proper locus of the transformational analysis is perhaps not the lexical stratum but the morpholexical stratum, that is, both lexotactics and morphotactics in combination. Be this as it may, if word fragments move there is no question that they cannot be phrases; they must be heads. [16] Now, there is a small class of verbs that have a peculiar defect. One of these verbs is *uraufführen* (to give the first performance of, to premiere).

(3.95) *Er hat diese Symphonie* ***uraufgeführt****.*
He has given the first performance of this symphony.

(3.96) **Er* ***uraufführte*** *diese Symphonie.*
**Er* ***aufführte*** *diese Symphonie* ***ur****.*
**Er* ***führte*** *diese Symphonie* ***urauf****.*
He gave the first performance of this symphony.

(3.97) *Er diese Symphonie* ***uraufführen*** *– lächerlich!*
He to give the first performance of this symphony – ridiculous!

(3.98) *Ich erinnere mich daran, als er diese Symphonie* ***uraufführte****.*
I recall when he gave the first performance of this symphony.

It is at first sight rather hard to say why the sentence (3.95) is fine, but its past tense variant (3.96) is not. Morphologically, nothing is wrong with *uraufführte*, as shows (3.98). It can also not be the fact that it is in a main clause, as shows (3.97). Rather, it seems that *uraufführen* is caught between two requirements. *Aufführen* is a verb that must split in the way shown above. Thus, in the main clause *auf-* and *führen* must be in different positions. On the other hand, *ur-* does not want to split. It requires the integrity of the verb. Thus, because *auf-* never moves as of these prefixes, and the prefix *ur-* is stuck with *auf-* but needs the verbal head, as a consequence the entire complex cannot move, neither partly nor as a whole. It is then verbs such as *uraufführen* which let us exceptionally see where the element has been at d-structure. Ordinarily we must always reckon with the possibility

[16] Unfortunately, this requires still deeper argumentation. For once we are talking about morphemes and not lexemes we have to offer an analysis of that part of the lexeme that is being moved. For notice that in the combination preposition + verb we must be able to say that the verb is the head of the construction if we are to argue that what is at hand is head movement. For that to be so, the terminology must be extended to the morphotactic stratum. This has been done by some morphologists; however, in X-bar-syntax it is assumed that on the lexical level there is only adjunction and no room for structures *head+complement*. Yet, as a quick review of morphological facts show this is not a tenable view. We must leave matters undecided and vague here.

that where we see constituents is not where they have been at D-structure. In this case we have a little window to D-structure.

Now that we have clarified the origin of the verbal head we must also find out what the target is and what the resulting structure after movement is. Notice that if a verbal head moves by substitution into a position, it can only be a position of same type, a position of a verbal head. But the evidence presented under (3.90) makes this unlikely. Thus, the verbal head cannot substitute. It is therefore assumed that it adjoins to a head position. This has several consequences, among other that the head does not strictly c-command its trace. On the other hand, in the derived position it does not project its own category features, but it allows the original head to do that. Thus adjunction does not disturb the local constitution of the tree. This given we have to find out what head position the verb adjoins to. To answer that we look at wh-phrases.

(3.99)	*Wer* ***hat*** *was gestern unterschrieben?* *Who has signed what yesterday?*
(3.100)	**Wer was* ***hat*** *gestern unterschrieben?* *Who what has yesterday signed?*
(3.101)	****Hat*** *wer was gestern unterschrieben?* *Has who what yesterday signed?*

This data suggests that the finite verb adjoins to the complementizer head. Given that wh-phrases are in specifier of the complementizer phrase this will readily explain the second position of the finite verb. However, notice that we must then also assume that the first argument of the sentence is in this position, too. [17] To make this work, two additional assumptions are needed. First, the complementizer position may not be left empty. Second, an overt complementizer blocks adjunction of the verb. The first will make sure that the verb has reason to move into second place, and that the process is obligatory when there is no complementizer. On the other hand, an overt complementizer fills the vacant position, and the reason for the verb to move adjoin to it is lost. This explains why subordinate clauses show SVO structure. The claims made seem stipulative, but a morphological explanation may be found in the case of the ban to adjunction. As regards the first requirement, we must assume that it is an idiosyncrasy of Germanic languages. But here, too, more can be said. At this point, however, we leave the discussion.

[17] Unless, of course we postulate more functional heads with more positions to fill than just one. But the latter leaves us with the uncomfortable problem to state which of these positions must be empty when another is filled, and a lot more.

Chapter 4

Nearness

In this chapter we introduce the central theme of this book, namely *nearness*. Starting with the observation that syntactic processes never operate on the entire tree but work within certain neighbourhoods, we will develop a language for talking about nearness in trees.

4.1 The Local Character of Syntactic Processes

In the previous chapter we have sought to reduce the rule schemata as much as possible. The idea behind this was the conception that we could factor out several independent rule schemata, the conjuration of which would be the ones that we observe in natural language. Thus, we hope that ideally we are left with a skeleton (the X-bar schema) and that an individual rule is obtained by providing more details coming from various subcomponents, which are yet to be identified. This is pretty much the concept of the theory of Government and Binding, GB. It has identified various subtheories, each dealing with a special phenomenon of language. These subtheories were factored out from the phrase-structure schema in order to gain a principled understanding of the possibilities and limits of natural language. Ideally, each of these subtheories would be a rather small bundle of constraints (called *module*) and that the maze of constructions will be the effect of superimposing all these constraints.

We have made some progress with respect to identifying these subtheories. We have seen that there is *agreement*, *selection*, *case*, *movement* etc. Now, however, we will attempt at reviewing the basic properties of these individual subtheories, or modules. What is interesting is that they can all be rendered in a uniform way as nearness statements. A *nearness statement* is a statement of the form *if a node x of category α then all nodes in the R-domain of x are of category β* or a statement of the form *each node x of category α*

has exactly one node y of category β within its R-domain. Here, we have made use of categories and so-called *R-domains*. The latter are based on a binary relation R on (labelled, ordered) trees, which depends on the nature of the syntactic process, that is, the subtheory. It is the question of what counts as a suitable and natural relation R that interests us here. Notice also that the general philosphy is that we start with the mere classificatory part of language, assume that there is a rule skeleton and a set of correlations with respect to certain features such as agreement features, case features etc. Thus, these features are not spelled out in tandem with the phrase structure rules but are kept strictly apart. We can identify agreement features, or selectional properties indepently of their exact behaviour, but the test of their existence lies — of course — in the correlations that we can establish. So, when we see that some features of the subject NP correlate with certain features of the V(P) then this calls for a theory of this correlation; this correlation comprises two things, namely a look-up table of what corresponds with what (we will see that this is a nontrivial issue) plus a description of the structural relationship that nodes must enter if they are forced to correlate in the given way.

Before we present this theory in its abstract form, however, let us look at some motivating cases. The first complex is subject-verb agreement. Agreement consists of two ingredients. One is the bare correlation table, namely that a singular third person noun phrase correlates with verb plus ending /*s*/ in the indicative present tense form (which includes the auxiliary in the perfect tense), while any other noun–phrase demands the non-suffixed form. The other is a specification of which subject agrees with what verb. When we say that subject and verb agree, we mean to say that subject and verb of the same sentence agree. Not any pair subject–verb is required to agree, but just those that are contained in the same sentence. This accounts for the grammaticality of the sentences (4.1) and (4.2) and the ungrammaticality of (4.3) and (4.4).

(4.1) *William believed that the firemen* ***have*** *rescued the president.*
(4.2) *The firemen believed that William* ***has*** *rescued the president.*
(4.3) **William* ***believe*** *that the president has been rescued.*
(4.4) **The firemen* ***believes*** *that the president has been rescued.*

These observations are in fact quite easy to make. However, formulating an exact principle turns out not to be so easy, because the story is complicated by many factors. Let us try to get at the idea, however. What we want to say is that the least node of category *sentence* above the subject is the least node of category *sentence* above the verb. However, for some reason that will become clear we will separate this into two nearness principles. We say that a node x *sentence*–commands another node y if the least node of category *sentence* properly dominating x also dominates y. Given that terminology, we can say that if x is the subject of a sentence, and x *sentence*–commands a node y that is a verb and y *sentence*–commands x, then x and y must agree. This statement just encodes the structural relationship that x and y must enter in order to be forced to agree, but does not say anything about what we mean by agreement. We should also say that there are languages in which the verb not only agrees with the subject, but also with the object (e. g. Basque, Georgian).

Another matter is *case*. We find here that the verb assigns case to its complements, or rather that it uses case and other properties to identify the relevant variable an argument is to be associated with. Moreover, in standard GB it is assumed that the verb has a single complement (if it has arguments at all) and that it assigns case to this complement, while the other arguments get their case through a different mechanism. We have adduced arguments against that view; we can nevertheless formulate a theory of case-assignment that is compatible with the view that case is assigned only under the tightest possible nearness-relationship, that of sisterhood. It seems that (with some exceptions, usually traded under the name ECM, for *exceptional case marking*) we can state that if a category α has a case to assign, then the potential receiver of case must be the sister of a node bearing the label α. This roundabout way of talking is necessitated by the fact that we want to separate the fact that an item induces a correlation from the fact that it assigns case. Thus, other than previously, where we associated with a word a (possibly empty) list of selected arguments — which in turn could be specified for case and must be discharged one at a time —, we now separate the selectional part from the structural part, allowing us to talk merely about selectional properties, and — separately — about the structural condition under which these properties become operative. For the sake of exposition, let us take the simplest case, that of a direct object. A verb that has a direct object, also assigns case to it, and in English this is accusative case. This is the selectional property of the verb. We say that it selects a direct object. Next we need to say where it expects to find that direct object. In English, it is assumed that it expects its object to be a sister. Thus the following data is accounted for.

(4.5) *Wayne did not raise this question.*
(4.6) **Wayne did not raise.*
(4.7) ?*Wayne did not raise yesterday this question.*

(4.6) is ungrammatical because there is no object even though the verb needs one. (4.7) is ungrammatical because there is no object that is sister to the verb.

Notice however that the predictions of this requirement must be calculated against a series of loopholes that the theory of transformational grammar provides. We may violate the requirement of sisterhood simply because the object is entitled to move into specifier of the complementizer position, for example when it is a question–phrase.

(4.8) *Which question did Wayne not raise (yesterday)?*

To account for the acceptability of (4.8) we assume that movement leaves a trace, and that the trace provides enough local structure so that the selectional requirement of the verb is fulfilled. In other words, the object trace is assumed to be a noun phrase with case accusative etc., but phonetically empty. Using this, we can also account for the apparent possibility to have (4.7), which does not seem downright ungrammatical. We can assume that *this question* has been extraposed to the right, and adjoined to some higher node. This is manifested also in the fact that we need to use a special intonation contour to make this sentence alright, setting apart somewhat the adverb *yesterday*.

Thus, the neighbourhood in which the verb expects the direct object, is the neighbourhood of the most immediate constituent containing that verb; in technical terms, we say that the verb *c-commands* the object. That is to say, since the object cannot be the verb nor contain it, nor be contained in it, this is the same as to require that the direct object be sister to the verb. We will later return to these subtleties. Now let us look at the problem of assigning the correct case feature in general. For even if we have settled the question of what case feature the complement of the verb has, this does not mean the issue is fully settled. First, notice that from the perspective that we developed in the earlier chapters, to say that the object noun-phrase has accusative case is not what we can observe. We can only observe that the elements of which that noun–phrase consists agree in case — with noteworthy exceptions. So, consider the following data.

(4.9) *Cäsar befahl* ***den Angriff***.
Cesar ordered the attack.

(4.10) *Cäsar befahl* ***die Zerstörung der Stadt***.
Caesaer ordered the destruction of the city.

We see that when the noun–phrase consists of several words as in (4.9), they all agree in case. However, there is a conflict that arises as a possibility. Suppose that a verb selects a complement, and that complement contains a case assigner γ whose complement is α. Then what should be the case of α if the verb assigns a different case as does γ? There we observe in German (and many other languages) that the closest assigner, in this case γ, wins. This is exemplified in (4.10). The verb *befahl* assigns accusative case to its complement whose head is the noun *Zerstörung* which is itself a case assigner. It assigns genitive case, however. The noun–phrase *der Stadt* appears in the genitive case, so does not agree in case with the complement *Zerstörung*.

The general line of thought is that closer assigners intervene in the relationship. So in principle we say that γ, being a case assigner, assigns case to the entire complement β but any intervening case assigner blocks the appearance of case. There are two immediate questions. First, what justifies us saying that the case assigner assigns case to the entire complement if this appears to be contrary to fact? And secondly, what allows us to say that the closest case assigner wins? We answer the second question first, and this will also throw some light on the first. There are languages in which we can have several case endings. Consider the sentence from (Blake, 1994).

(4.11)

Maku-ntha	*yulawu-jarra-ntha*	*yakuri-naa-ntha*
woman OBL	catch-TAST-OBL	fish-MABL-OBL
dangka-karra-nguni-naa-ntha		*mijil-nguni-naa-ntha*
man-GEN-INST-MABL-OBL		net-INST-MABL-OBL

‘The woman must have caught fish with the man’s net.’

In this example, OBL stands for *oblique*, PAST for *past tense*, *mabl* means *modal ablative*, GEN is *genitive*, and INST is *intrumental*. What is observed is that we can stack up to four case suffixes, that all the words are suffixed by *-ntha*, meaning oblique. So rather than one case assigner winning over another, each case assigner is allowed to contribute a suffix, so to speak, where the innermost suffix is the one corresponding to the closest assigner.

Another remarkable phenomenon is found in Old Georgian. A genitive attribute if following a noun phrase is marked for genitive in addition to receiving the case ending of the entire noun phrase. For example

(4.12)	*sarel-ita*	*man-isa-jta*
	name-INST	father-GEN-INST
	'with father's name'	

Again the innermost case corresponds to the case assigned by the closest assigner. Thus it is not always the case that we have a blockade of case assigning properties. And this answers the first question as well. We may assume at least in *all* the constructions shown above that the case assigner assigns case to the entire constituent; languages differ, however, in the way the case information gets realized if various case assigners compete. In fact, what is known as *abstract case* can be assimilated to this solution as well. We say that even if in English there is no overt case marking, there is case assigned but it is not realized. So while German, Latin, Finnish and many other languages realize 1 case, Kayardild 4 cases, [1] Georgian any number, English realizes none.

4.2 Command Relations

Before we continue with the motivating examples and develop some formal terminology. We start with the notion of command relation defined by CHRIS BARKER and GEOFFREY PULLUM in (Barker and Pullum, 1990). Let $\mathfrak{T}$ be a tree with underlying set T. A command relation is going to be a binary relation on T. Given such a binary relation R we write $xR := \{y : xRy\}$ and call it the *R-domain* of x. The first requirement that we pose on command relations is that xR must always be a constituent. This means that R can be represented by a function $f_R : T \to T$, which satisfies $xR = \downarrow f_R(x)$. We call f_R the *domain function* of R. Call a function $f : T \to T$ *strongly extensive* if for all x either x is the root and $f(x) = x$ or else $f(x) > x$. Call f *increasing* or *monotone* if for all x and y if $x \leq y$ then $f(x) \leq y$.

Definition 4.2.1 *A relation R on a tree is called a* **command relation** *if the R-domain of a node x is constituent, so that $xR = \downarrow f_R(x)$ for some function $f_R : T \to T$ which is strongly extensive and monotone.*

[1]This is an oversimplification; the cases cannot be iterated, that is, not cases of the same type. (Blake, 1994) speaks of four layers of case. But this anyway just an illustration. A detailed analysis still has to be done.

This explained in the following way. We are interested in the set of nodes with are related via some syntactic process with a given node x. Say, this process is case assignment. Primarily, this is a relation between certain constituents, say the verb and the direct object. However, there are reasons why we want to say that it extends to the constituent. One is that the property of having case is visible at the leaves of the tree only, because the overt material (the phonological string) can only attach there. Thus we must assume that the influence of x really extends to the leaves of that constituent. This is a purely observational criterion. We may say that the influence of x over y is mediated by a z which is directly influenced by x (the complement of x, say) and which by some other rule influences y. This is possible, though it does not invalidate the claim that x influences y. Second, we have previously noted that the elements that x influences do actually *not* form a constituent. There are several exceptions. First, a closer case assigner may block the influence of x. We have explained this by saying that it does not block this influence but adds another, which is the one that is visible. There are languages in which we can see elements being marked for double influence. Secondly, there is a usual exemption of x itself plus whatever is dominated by x or dominates x. It is awkward to say that x influences whatever it is itself made up of. However, the difficulties arise with respect to a particular interpretation of influence, namely when we consider the tree as being built bottom-to-top by successively merging trees into larger trees. Then as soon as x comes into existence it no longer influences its parts. Yet then it is plausible to say that x may influence its mother. Conversely, if we assume trees to be produced top-down (when produced by grammars) then x may well be seen to influence its parts, but not what itself is part of. As the trees here are static objects, neither view is right. We simply see all nodes as existing independently of each other and exerting influence onto each other. So we are taking here a purely descriptive attitude towards sentence structure. Finally, the exemption clauses also are relevant because they correspond to elements which are not ordered with respect to x. So, exempting the constituent as well as the position of x from the influence of x is the same as saying x must not overlap with elements of its domain.

Given a command relation R and a pair x, y such that $x\,R\,y$ we say that x is the *R-head* of y and that y is the *R-foot* of x. The R-feet of a node form a constituent. The R-heads for a given node y form an upper closed set, by monotonicity of the associated function. Domains may never shrink when we go up the tree. We say that x and y are *co–heads* if they have the same R-domain, and that they are *co–feet* if they are contained in the same R-domains. An *R-chamber* is a maximal set of co–heads, and a *cell* a maximal set of co-feet. Finally, x and y are *R-mates* if $x\,R\,y$ and $y\,R\,x$.

The smallest possible command relation is the one obtained by taking the following function. $f_c(x)$ is the mother of x if x is not the root, and the root otherwise. The relation thus generated is called *idc-command*. Standardly, authors call idc-command *c-command*. However, there are various definitions of c-command, differing because c-command was invented to do justice to binding facts, and with the theory trying to accommodate the data the definition of c-command has been frequently altered. This is why, following

Barker and Pullum, we have chosen idc-command. Another special variety of command relations are the *tight* relations.

Definition 4.2.2 *A command relation is called **tight** if for $y < f_R(x)$ we have $f_R(y) \leq f_R(x)$.*

This is a purely technical definition whose meaning will become clear later on. There is a particular way of understanding this definition which reveals something about the property of tight relations. Say that we move inside the tree in the following way. We start at z_0 and are allowed to move inside the R-domain of z_0, but not to the generating node of the constituent, which is $f_R(z_0)$. Then from that new point, z_1, we may move into the R-domain of z_1. Then if the command relation is tight we can never leave the original domain, $f_R(z_0)$.

Tight relations can be characterized in a different way as follows. Select a set $S \subseteq T$. Say that x *S-commands* y if for every node $z \in S$ properly dominating x, $z \geq y$. This relation will be denoted by S. Thus, the associated function is computed as follows. Put $g_S(x) := r$ if there exists no $z \in S$ such that $z > x$. Otherwise, let $g_S(x)$ be the least node in S properly above x. Defining $min(U)$ to be the minimal element if it exists, and $= r$ else, we can define $g_S(x)$ simply by

$$g_S(x) := minS \cap \{y : y > x\}.$$

This defines a command relation, as is straightforward to check. Now a command relation is *fair* if its associated function is of the form g_S for some $S \subseteq T$. Idc-command is tight, and we can easily see that its associated function is g_T, T the entire set of nodes.

Theorem 4.2.3 *A command relation R is fair iff it is tight.*

Proof. Suppose R is fair. Then its associated function has the form g_S. Now let $y < g_S(x)$. Then two cases arise. Either $g_S(x)$ is the root, in which case $g_S(y) \leq g_S(x)$ holds anyway. Or it is not the root, and then $g_S(x) \in S$. Consequently, $g_S(y) \leq g_S(x)$. Now assume conversely that R is tight. Put $S := \{f_R(x) : x \in T\}$. Then we have to show that $f_R(x)$ is the least node of the form $f_R(y)$ strictly above x, unless x is the root. Suppose, however, that we have $x < f_R(y) < f_R(x)$. Then by tightness, $f_R(x) \leq f_R(y)$, contradiction. ⊣

Tight relations have an important property; even when the structure of the tree is lost and we know only P we can recover g_P and $<$ to some extent. Notice namely that if $\mathsf{P}_x \neq T$ then $g_P(x)$ is the unique y such that $y \in \mathsf{P}_x$ but the P-domain of y is larger than the P-domain of x. We can then exactly say which elements are dominated by y: exactly the elements of the P-domain of x. By consequence, if we are given T, the root r and we know the idc-command domains, $<$ can be recovered completely. This is of relevance to syntax because often the tree structures are not given directly but are recovered using domains. Let f, g be strongly extensive functions; then define

$$\begin{array}{rcl}(f \sqcup g)(x) & := & max\{f(x), g(x)\} \\ (f \sqcap g)(x) & := & min\{f(x), g(x)\} \\ (f \circ g)(x) & := & f(g(x))\end{array}$$

Since $f(x), g(x) \geq x$, that is, $f(x), g(x) \in \uparrow x$ and since $\uparrow x$ is linear, the maximum and minimum are always defined. Clearly, with f and g strongly extensive, $f \sqcup g$, $f \sqcap g$ and $f \circ g$ are also strongly extensive.

Lemma 4.2.4 $f_{R \cup S} = f_R \sqcup f_S$. $f_{R \cap S} = f_R \sqcap f_S$.

Proof. $z \leq f_{R \cup S}(x)$ iff $x(R \cup S)z$ iff either xRz or xSz iff either $z \leq f_R(x)$ or $z \leq f_S(x)$ iff $z \leq max\{f_R(x), f_S(x)\}$. Analogously for intersection. In remains to be shown that the intersection and union is monotone. Now let $x \leq y$. Then by assumption $f_R(x) \leq f_R(y)$ and $f_S(x) \leq f_S(y)$. Then $f_R(x) \sqcap f_S(y) \leq f_R(y), f_S(y)$ and so $f_R(x) \sqcap f_S(x) \leq f_{R \cap S}(y)$. Similarly for union. ⊣

Theorem 4.2.5 *For any given tree $\mathbb{T}$ the command relations over $\mathbb{T}$ form a distributive lattice $\mathfrak{Cr}(\mathbb{T}) = \langle Cr(\mathbb{T}), \cap, \cup \rangle$.*

Proof. By the above lemma, the command relations over $\mathbb{T}$ are closed under intersection and union. Distributivity automatically follows since lattices isomorphic to lattices of sets with intersection and union as operations are always distributive. ⊣

Proposition 4.2.6 $g_{P \cup Q} = g_P \sqcap g_Q$. *Hence tight relations over a tree are closed under intersection. They are generally not closed under closed union.*

Proof. Let $P, Q \subseteq T$ be two sets upon which the relations P and Q are based. Then the intersection of the relations, $\mathsf{P} \cap \mathsf{Q}$, is derived from the union $P \cup Q$ of the basic sets. Namely, $g_{P \cup Q}(x) = min\{y : y \in P \cup Q, y > x\} = min\{min\{y : y \in P, y > x\}, min\{y : y \in Q, y > x\}\} = min\{g_P(x), g_Q(x)\} = (g_P \sqcap g_Q)(x)$. To see that tight relations are not necessarily closed under union take the union of NP-command and S-command. If it were tight, the nodes of the form $g(x)$ for some x define the set on which this relation must be based. But this set is exactly the set of bounding nodes, which defines Lasnik's kommand. The latter, however, is the intersection, not the union of these relations. ⊣

The consequences of this theorem are the following. The tight relations form a sub-semilattice of the lattice of command relations; this semi-lattice is isomorphic to $\langle 2^{int(\mathbb{T})}, \cup \rangle$. Although the natural join of tight relations is not necessarily tight, it is possible to define a join in the semi-lattice. This operation is completely determined by the meet-semilattice structure, because this structure determines the partial order of the elements which in turn defines the join. In order to distinguish this join from the ordinary one we write it as $\mathsf{P} \bullet \mathsf{Q}$. The corresponding basic set from which this relation is generated is the set $P \cap Q$; this is the only choice, beacuse the semilattice $\langle 2^{int(\mathbb{T})}, \cup \rangle$ allows only one extension to a lattice, namely $\langle 2^{int(\mathbb{T})}, \cup, \cap \rangle$. The notation for associated functions is the same as for the relations. If g_P and g_Q are associated functions, then $g_P \bullet g_Q = g_{P \cap Q}$ denotes the associated function of the (tight) join.

Consider the definition of the relational product

$$R \circ S = \{\langle x, z \rangle : (\exists y)(xRySz)\}$$

Then $f_{R \circ S} = f_S \circ f_R$ (with converse ordering!). For a proof consider the largest z such that $x(R \circ S)z$. Then there exists a y such that $xRySz$. Now let $\tilde{y}$ be the largest y such that xRy. Then not only $xR\tilde{y}$ but also $\tilde{y}Sz$, since S is monotone. By choice of $\tilde{y}$, $\tilde{y} = f_R(x)$. By choice of z, $z = f_S(\tilde{y})$, since $f_S(\tilde{y}) > z$ would contradict the maximality of z. In total, $z = (f_S \circ f_R)(x)$ and that had to be proved.

From the theory of binary relations it is known that $\circ$ distributes over $\cup$, that is, that we have $R \circ (S \cup T) = (R \circ S) \cup (R \circ T)$ as well as $(S \cup T) \circ R = (S \circ R) \cup (T \circ R)$. But in this special setting $\circ$ also distributes over $\cap$.

Proposition 4.2.7 *Let R, S, T be command relations. Then $R \circ (S \cap T) = (R \circ S) \cap (R \circ T)$, $(S \cap T) \circ R = (S \circ R) \cap (T \circ R)$.*

Proof. Let $x(R \circ (S \cap T))z$, that is, $xRy(S \cap T)z$, that is, $xRySz$ and $xRyTz$ for some y. Then, by definition, $x(R \circ S)z$ and $x(R \circ T)z$ and so $x((R \circ S) \cap (R \circ T))z$. Conversely, if the latter is true then $x(R \circ S)z$ and $x(R \circ T)z$ and so there are y_1, y_2 with xRy_1Sz and xRy_2Tz. With $y := max\{y_1, y_2\}$ we have $xRy(S \cap T)z$ since S, T are monotone. Thus $x(R \circ (S \cap T))z$. Now for the second claim. Assume $x((S \cap T) \circ R)z$, that is, $x(S \cap T)yRz$ for some y. Then xSy, xTy and yRz, which means $x(S \circ R)z$ and $x(T \circ R)z$ and so $x((S \circ R) \cap (T \circ R))z$. Conversely, if the latter holds then $x(S \circ R)z$ and $x(T \circ R)z$ and so there exist y_1, y_2 with xSy_1Rz and xTy_2Rz. Put $y := min\{y_1, y_2\}$. Then xSy, xTy, hence $x(S \cap T)y$. Moreover, yRz, from which $x((S \cap T) \circ R)z$. $\dashv$

Definition 4.2.8 *A **distributoid** is a structure $\mathfrak{D} = \langle D, \cap, \cup, \circ \rangle$ such that (1) $\langle D, \cap, \cup \rangle$ is a distributive lattice, (2) $\circ$ an associative operation and (3) $\circ$ distributes both over $\cap$ and $\cup$.*

Theorem 4.2.9 *The command relations over a given tree form a distributoid denoted by $\mathfrak{Dis}(\mathbb{T})$.* $\dashv$

The fact that distributoids have so many distributive laws means that for composite command relations there are quite simple normal forms. Namely, if $\mathfrak{R}$ is a command relation composed from the $R_1, \dots, R_n$ by means of $\cap, \cup$ and $\circ$, then we can reproduce $\mathfrak{R}$ in the following simple form. Call $\mathfrak{C}$ a **chain** if it is composed from the R_i using only $\circ$. Then $\mathfrak{R}$ is identical to an intersection of unions of chains, and it is identical to a union of intersections of chains. Namely, by (3), both $\cap$ and $\cup$ can be moved outside the scope of $\circ$. Moreover, $\cap$ can be moved outside the scope of $\cup$ and $\cup$ can be moved outside the scope of $\cap$.

Theorem 4.2.10 (Normal Forms) *For every* $\mathfrak{R} = \mathfrak{R}(R_1, \dots, R_n)$ *there exist chains* $\mathfrak{C}_i^j = \mathfrak{C}_i^j(R_1, \dots, R_n)$ *and* $\mathfrak{D}_i^j = \mathfrak{D}_i^j(R_1, \dots, R_n)$ *such that* $\mathfrak{R} = \bigcup_i \mathfrak{I}_i$ *with* $\mathfrak{I}_i = \bigcap_j \mathfrak{C}_i^j$ *and* $\mathfrak{R} = \bigcap_j \mathfrak{U}_j$ *with* $\mathfrak{U}_j = \bigcup_i \mathfrak{D}_i^j$. $\dashv$

From the linguistic point of view, tight relations play a key role because they are defined as a kind of topological closure of nodes with respect to the topology induced by the various categories. (However, this analogy is not perfect because the topological closure is an idempotent operation while the domain closure yields larger and larger sets, eventually being the whole tree.) It is therefore reasonable to assume that all kinds of linguistic CRs be defined using tight relations as primitives. Indeed, (Koster, 1986) argues for quite specific choices of fundamental relations, which will be discussed below. It is worthwile to ask how much can be defined from tight relations. This proves to yield quite unexpected answers. Namely, it turns out that union can be eliminated in presence of intersection and composition. We prove this first for the most simple case.

Lemma 4.2.11 *Let* g_P, g_Q *be the associated functions of tight relations. Then*

$$g_P \sqcup g_Q = (g_P \circ g_Q) \sqcap (g_Q \circ g_P) \sqcap (g_P \bullet g_Q).$$

Proof. First of all, since $g_P, g_Q \leq g_P \circ g_Q, g_Q \circ g_P, g_P \bullet g_Q$ we have $g_P \sqcup g_Q \leq (g_P \circ g_Q) \sqcap (g_Q \circ g_P) \sqcap (g_P \bullet g_Q)$. The converse inequation needs to be established. There are three cases for a node x. (i) $g_P(x) = g_Q(x)$. Then $(g_P \sqcup g_Q)(x) = g_{P \cap Q}(x) = (g_P \bullet g_Q)(x)$, because the next P-node above x is identical to the next Q-node above x and so is identical to the next $P \cap Q$-node above x. (ii) $g_P(x) < g_Q(x)$. Then with $y = g_P(x)$ we also have $g_Q(y) = g_Q(x)$, by tightness. Hence $(g_P \sqcup g_Q)(x) = (g_Q \circ g_P)(x)$. (iii) $g_P(x) > g_Q(x)$. Then as in (ii) $(g_P \sqcup g_Q)(x) = (g_P \circ g_Q)(x)$. $\dashv$

The next case is the union of two chains of tight relations. Let $\mathfrak{g} = g_m \circ g_{m-1} \dots \circ g_1$ and $\mathfrak{h} = h_n \circ h_{n-1} \dots \circ h_1$ be two associated functions of such chains. Then define a **splice** of $\mathfrak{g}$ and $\mathfrak{h}$ to be any chain $\mathfrak{k} = k_\ell \circ k_{\ell-1} \dots \circ k_1$ such that $\ell = m + n$ and $k_i = g_j$ or $k_i = h_j$ for some j and each g_i and h_j occurs exactly once and the order of the g_i as well as the order of the h_i in the splice is as in their original chain. So, the situation is comparable with shuffling two decks of cards into each other. A **weak splice** is obtained from a splice by replacing some number of $g_i \circ h_j$ and $h_j \circ g_i$ by $g_i \bullet h_j$, the least tight relation containing both g_i and h_j. In a weak splice, the shuffling is not perfect in the sense that some pairs of cards may be glued to each other. If $\mathfrak{g} = g_2 \circ g_1$ and $\mathfrak{h} = h_2 \circ h_1$ then the following are all splices of $\mathfrak{g}$ and $\mathfrak{h}$: $g_2 \circ g_1 \circ h_2 \circ h_1$, $g_2 \circ h_2 \circ g_1 \circ h_1$, $g_2 \circ h_2 \circ h_1 \circ g_1$. The following are weak splices (in addition to the splices, which are also weak splices): $g_2 \circ g_1 \bullet h_2 \circ h_1$, $g_2 \bullet h_2 \circ g_1 \bullet h_1$. A non-splice is $g_1 \circ h_2 \circ g_2 \circ h_1$, and $g_2 \bullet g_1 \circ h_2 \circ h_1$ is not a weak splice.

Lemma 4.2.12 *Let* $\mathfrak{g}$, $\mathfrak{h}$ *be two chains of tight relations (or their associated functions). Let* $\mathbf{wk}(\mathfrak{g}, \mathfrak{h})$ *be the set of weak splices of* $\mathfrak{g}$ *and* $\mathfrak{h}$*. Then*

$$\mathfrak{g} \sqcup \mathfrak{h} = \bigsqcap \langle \mathfrak{s} : \mathfrak{s} \in \mathbf{wk}(\mathfrak{g}, \mathfrak{h}) \rangle$$

.

Proof. As before, it is not difficult to show that $\mathfrak{g} \sqcup \mathfrak{h} \leq \bigsqcap \langle \mathfrak{s} : \mathfrak{s} \in \mathbf{wk}(\mathfrak{g}, \mathfrak{h}) \rangle$ because $\mathfrak{g}, \mathfrak{h} \leq \mathfrak{s}$ for each weak splice. So it is enough to show that the left hand side is equal to one of the weak splices in any tree for any given node. Consider therefore a tree $\mathbb{T}$ and a node $x \in T$. We define a weak splice $\mathfrak{s}$ such that $\mathfrak{s}(x) = max\{\mathfrak{g}(x), \mathfrak{h}(x)\}$. To this end we define the following nodes. $x_0 = x, y_0 = x$, $x_1 = g_1(x_0), h_1(y_0), \ldots, x_{i+1} = g_{i+1}(x_i), y_{i+1} = h_{i+1}(y_i), \ldots$. The x_i and the y_i each form an increasing sequence. We can also assume that both sequences are strictly increasing because otherwise there would be an i such that $x_i = r$ or $y_i = r$. Then $(\mathfrak{g} \sqcup \mathfrak{h})(x) = r$ and so for any weak splice $\mathfrak{s}(x) = r$ as well. So, all the x_i can be assumed distinct and all the y_i as well. Now we define z_i as follows. $z_0 = x$, $z_1 = min\{x_1, \ldots, x_m, y_1, \ldots, y_n\}, \ldots, z_{i+1} = min(\{x_1, \ldots, x_m, y_1, \ldots, y_n\} - \{z_1, \ldots, z_i\})$. Thus, the sequence of the z_i is obtained by fusing the two sequences along the order given by the upper segment $\uparrow x$. Finally, the weak splice can be defined. We begin with s_1. If $x_1 = y_1$, $s_1 = g_1 \bullet h_1$, if $x_1 < y_1$, $s_1 = g_1$ and if $x_1 > y_1$ then $s_1 = h_1$. Generally, for z_{i+1} there are three cases. First, $z_{i+1} = x_j = y_k$ for some j, k. Then $s_{i+1} = g_j \bullet h_k$. Else $z_{i+1} = x_j$ for some j, but $z_{i+1} \neq y_k$ for all k. Then $s_{i+1} = g_j$. Or else $z_{i+1} = y_k$ for some k but $z_{i+1} \neq x_j$ for all j; then $s_{i+1} = h_k$. It is straightforward to show that $\mathfrak{s}$ as just defined is a weak splice, that $z_{i+1} = s_i(z_i)$ and hence that $\mathfrak{s}(x) = max\{\mathfrak{g}(x), \mathfrak{h}(x)\}$. ⊣

The tight relations generate a subdistributoid $\mathfrak{Tgr}(\mathbb{T})$ in $\mathfrak{Dis}(\mathbb{T})$ members of which we call **tight generable**.

Theorem 4.2.13 *Each tight generable command relation is an intersection of chains of tight relations.* ⊣

4.3 The Distributoid Structure of Nearness Relations

We have previously studied the so-called tight generable relation. The reason for the interest in them is the fact that we can use the labels to define the relations schematically over the trees. Each boolean label a defines the relation of a-**command** on a fully labelled tree via the set of nodes of category **a**. This is the classical scenario; the label s defines s-command, the label $\mathsf{np} \cup \mathsf{cp}$ defines Lasnik's Kommand. And so forth. We denote the particular relation induced on $\langle \mathbb{T}, \ell \rangle$ by $\delta_{\mathbb{T}}(\mathsf{a})$. From this basic set of tight command relations we allow to define more complex command relations using the operations. To do this we have defined a constructor language that contains a constant a for each boolean label a and the binary symbols $\wedge$, $\vee$ and $\circ$. (Although we also use $\bullet$, we will treat it as an abbreviation; also, this operation is defined only for tight relations.) Since we assume the equations of distributoids, the symbols a generate a distributoid with $\wedge$, $\vee$, $\circ$, namely the so-called **free distributoid**. The map $\delta_{\mathbb{T}}$ can be extended to a homomorphism from this distributoid into $\mathfrak{Dis}(\mathbb{T})$. Simply put

$$\begin{array}{rcl} \delta_{\mathbb{T}}(\mathfrak{d} \wedge \mathfrak{e}) & = & \delta_{\mathbb{T}}(\mathfrak{d}) \cap \delta_{\mathbb{T}}(\mathfrak{e}) \\ \delta_{\mathbb{T}}(\mathfrak{d} \vee \mathfrak{e}) & = & \delta_{\mathbb{T}}(\mathfrak{d}) \cup \delta_{\mathbb{T}}(\mathfrak{e}) \\ \delta_{\mathbb{T}}(\mathfrak{d} \circ \mathfrak{e}) & = & \delta_{\mathbb{T}}(\mathfrak{d}) \circ \delta_{\mathbb{T}}(\mathfrak{e}) \end{array} \tag{4.15}$$

By definition, the image of $\mathfrak{d}$ under $\delta_{\mathbb{T}}$ is tight generable. Hence $\delta_{\mathbb{T}}$ maps all nearness terms into tight generable relations. With $\mathsf{np} \cup \mathsf{cp}$ being 1-node subjaceny (for English) we find that $(\mathsf{np} \cup \mathsf{cp}) \circ (\mathsf{np} \cup \mathsf{cp})$ is 2-node subjacency (see below). Using a more complex definition it is possible to define 0- and 1-subjacency in the barriers system on the condition that there are no double segments of a category. If we consider the power of subsystems of this language, e. g. relations definable using only $\wedge$ etc. the following picture emerges.

$$\begin{tikzcd}
& \{\circ, \wedge\} & \\
& \{\circ, \vee\} \arrow[u, no head] & \\
\{\circ\} \arrow[ur, no head] & & \{\vee, \wedge\} \arrow[ul, no head] \\
& \{\wedge\} \arrow[ul, no head] \arrow[ur, no head] &
\end{tikzcd} \tag{4.16}$$

This follows mainly from Theorem 4.2.13 because the map δ is by definition into the distributoid $\mathfrak{Tgr}(\mathbb{T})$ of tight generated command relations. Moreover, $\wedge$ alone does not create new command relations, because of Proposition 4.2.6. Each of the inclusions is proper as is not hard to see. So, $\vee$ does not add definitional strength in presence of $\circ$ and $\wedge$; although things may be more perspicuously phrased using $\vee$ it is in principle eliminable.

By requiring command relations to be intersections of chains we would therefore not express a real restriction at all.

It is important to realize that the formal operations on domains are not just mathematical games, but that they are linguistically meaningful. A particular reason is that these structures allow for what may be called *internalizations of definitions*. To exemplify this consider the case when a command relation R is defined in the following way from two commmand relations S and T.

(4.17) xRy iff xSy *and* xTy

This definition involves in addition to known symbols also the logical word *and*. If grammatical processes are regulated by domains we may ask ourselves whether or not it is possible to make the word *and* disappear. In the present case this is easy. Just let R be the intersection $S \cap T$ and the above is automatically fulfilled. This strategy to replace a complex definition of a domain by a single domain that encodes this definition we call **internalization**. As it stands, we have been able to internalize *and* by '$\cap$'; the reason for this is that command relations are closed under intersection and thus R — defined in this way — is again a command relation. By similar arguments we see that a definition of R via

(4.18) xRy iff xSy *or* xTy

yields $R = S \cup T$ and thus *or* is internalized by '$\cup$'. One may be inclined to think that this is just a formal device. But it does clarify certain issues, and is not a completely arbitrary affair. There are, for example, definitions that resist internalization. This means that they lead to non-domains and should for that reason always looked at with suspicion. To take a real example, consider the conditions [A] and [B] of the binding theory.

Condition [A] An anaphor must be bound inside its governing category.

Condition [B] A pronoun must be free in its governing category.

If we assume that we know how to define the set gc of governing categories the two conditions can be accurately rephrased as follows.

Condition [A] If x is an anaphor, then x gc-commands its antecedent.

Condition [B] If x is a pronoun, then x does not gc-command any of its antecedents.

Condition [B] is exactly the opposite or boolean negation of Condition [A]. Thus the 'domain' in which a pronoun may find an antecedent according to this definition is exactly the complement of the domain for [A], namely gc-command. However, it is easily seen that such a relation can never qualify as a command relation in our sense since it is not a constituent. Maybe this is the reason for the rather ill-fated history of Condition [B]. It exemplifies an instance of a definition of the type

(4.19) xRy iff *not* xSy.

Such a definition is not good as it does not define a command relation; equivalently, we may classify the notion of boolean negation by saying that *not* is not internalizable. Similarly, a definition

(4.20) xRy iff xSy *implies* xTy

fails to yield a command relation even if both S and T are CRs. Yet there are definitions that do correspond in some way to negation and implication, which can also be internalized. For example, let R be the largest command relation such that $S \cap R \subseteq T$; then this can be checked to be a command relation and internalizes some notion of implication, namely intuitionistic implication. Composition arises from a form of mediated command. Suppose we offer the following definition of R:

(4.21) xRy iff *for some* z xSz *and* zTy

Then x R-commands y iff there is a mediating element T-commanding y which is itself S-commanded by x. Then R is the *composition* of S and T and denoted by $S \circ T$. There are nontrivial cases which motivate the necessity of introducing composition. They arise in the theory of movement. One of the characteristic advantages of GB was that it could explain certain failures of extraction by the fact that movement from the D-structure position is not a one-step process, but must proceed via a sequence of intermediate steps. On the one hand, a *wh*–word can move from arbitrarily deep to the main clause as in (4.22), but on the other hand any intervening *wh*–word eliminates this possibility.

(4.22) *Who*$_1$ *do you wonder Mary wanted John to get hold of* t_1*?*
(4.23) **Who*$_1$ *do you wonder when*$_2$ *Mary wanted John to get hold of* $t_1 t_2$*?*

The explanation rests on two assumptions. The first is that a *wh*–element can only move a fixed distance per step. The second is that when moved, it can only be moved to specific places. The third is that these places are limited in number. To be precise, a *wh*–word has been assumed to only have the choice to fill the position of the specifier of the complementizer phrase. Generally, languages allow only for one such position per sentence. This was referred to as the *doubly–filled comp filter*. Let us not be concerned with that history, however. Crucially, a *wh*–word, wherever it originated, could move to the next specifier of a complementizer phrase. If that position was filled, either by an overt element or by a trace, then that movement was blocked. What the theory did not explain was why it was possible for a *wh*–element to stay if it could not move, and why on the other hand at least one of them had to move. We have spoken about that earlier. Let us here try to see whether we can capture the restrictions on the movement of *wh*–elements. There are clearly two. One is the typical restriction that the antecedent c-commands its trace, which we will for the moment equate with the restriction that the antecedent idc-commands its trace. The other, more interesting restriction is the requirement that the antecedent be not higher than the next complementizer phrase. This looks like a tight domain, but it is not. It can easily be demonstrated that a tight domain does not allow to be escaped by successive movement. In fact, the description just given is defective because if the *wh*-phrase is already in the position of a specifier of a complementizer phrase then instead of *next complementizer phrase* we should say *second next complementizer phrase*. This is indeed a command relation, as can easily be checked. But is it tight generated? Here we have to resort to a trick. Let cp stand for the label *is a complementizer phrase*. Its negation is $-\mathsf{cp}$. If a *wh*–phrase is not in the specifier of cp then the next phrase containing it is a $-\mathsf{cp}$. Hence, the following is a movement domain for *wh*–movement.

(4.24) $-\mathsf{cp} \circ \mathsf{cp}$

4.4 The Koster Matrix

The preceding chapter has been rather formal, just introducing notions and proving results. Now we want to connect this with syntactic theory. There have been some attempts within Government and Binding to approach syntax from the perspective of nearness domains. The most elaborate proposal is due to (Koster, 1986). The principal idea is that natural languages specify a limited range of locality restrictions, and that syntactic coercion between items in a sentence can take place only within such domains. Moreover, languages differ with respect to the nature of these domains, which follow a standard pattern, which we will call the *Koster Matrix*. We will outline this theory because it is the closest in spirit to the nearness analysis of grammar that we are going to propose. As Koster and many others have observed, grammatical relations are typically relations between a dependent element δ and an antecedent α.

$$\cdots \quad \underbrace{\alpha \quad \cdots \quad \delta}_{R} \quad \cdots$$

Four conditions are put on such configurations.

a. obligatoriness

b. uniqueness of the antecedent

c. c-command of the antecedent

d. locality

If these conditions are met then this relation has the effect

share property

This has to be understood as follows. (a.) and (b.) express nothing but that δ needs one and only one antecedent. This antecedent, α, must c-command δ. Finally, (d.) states that α must be found in some local domain of δ. This domain is language specific as well as specific to the syntactic construction, i. e. the category of δ and α. Likewise, the property to be shared depends on the category of α and δ. The locality restriction expresses that α is found within the R-domain of δ. This relation R is in the unmarked case defined as follows.

Definition 4.4.1 *α is **locally accessible**^{1} to δ if $\alpha \leq \beta$, where β is the least maximal projection containing δ and a governor of δ.*

In (Koster, 1986) it is assumed that greater domains are formed by licensed extensions. These extensions are marked constructions; while all languages agree on the local accessibility^1 as the minimal domain within which antecedents must be found, larger domains may also exist but their size is language and construction specific. Nevertheless, the variation is limited. There are only three basic types, namely *locally accessible^i* for $i = 1, 2, 3$.

Definition 4.4.2 *α is **locally accessible**^{2} to δ if $\alpha \leq \beta$, where β is the least maximal projection containing δ, a governor for δ and some opacity element ω. α is **locally accessible**^{3} to δ if there is a sequence β_i, $1 \leq n$, such that β_1 is locally accessible^2 from δ and β_{i+1} is locally accessible^2 from β_i.*

The opacity elements are drawn from a rather limited list. Such elements are *tense*, *mood* etc. A well-known example are Icelandic reflexives whose domain is the smallest indicative sentence. For an extensive study see (Koster and Reuland, 1991).

The local accessibility relations certainly are command relations in our sense. The real problem is whether they are definable using primitive labels of the grammar. In particular the recursiveness of the third accessibility condition makes it unlikely that we can find a definition in terms of $\wedge, \vee, \circ$. Yet, if it were really an arbitrary iteration of the second accessibility relation it would be completely trivial, because any iteration of a command relation over a tree is the total relation over the tree. Hence, there must be something non-trivial about this domain; indeed, the iteration is stopped if the outer β is ungoverned. This is the key to a non-iterative definition of the third accessibility relation.

Let us assume for simplicity that there is a single type of governors denoted by gov and that there is a single type of opacity element denoted by opy. The first hurdle is the clarification of *government*. Normally, government requires a governing element, i. e. an element of category gov that is close in some sense. How close, is not clarified in (Koster, 1986). Clearly, closeness cannot be accessibility^1, for then the definition would be circular. It must be an even smaller domain. Let us agree that a node has the property $\boxminus\mathsf{x}$ if *all of its sisters is of category x*, so that $\boxminus - \mathsf{x}$ means that none of the sisters is of category x. (For now, $\boxminus$ is just a device to describe certain sets of nodes. Later we shall see that it can be given a logical interpretation as a modal operator as well.) With this definition, being governed is equal to being of category $- \boxminus -\mathsf{gov}$. Notice that this does not exclude that there are several governors. Assuming binary branching, however, the governor is indeed unique, given that the relation *sister of* is irreflexive. Moreover, another constructor is introduced, namely $\diamondsuit$. A node is of category $\diamondsuit\mathsf{x}$ if *one of the immediate daughters is x*. These constructors will play an important role later on, so it is useful to study the definitions of the domains carefully. We will assume that the opacity element must be in IDC-command relation to δ. This is a simplifying assumption. The third assumption we make is that governors are heads, that is, they are not decomposable further. None of the assumptions is necessary, and if they fail one could adapt the definitions accordingly.

We are now ready to define the three accessibility relations, which we denote by LA^1, LA^2 and LA^3.

$$\begin{array}{rcr}
\mathsf{LA}^1 & = & \diamondsuit\mathsf{gov} \bullet \mathsf{bar{:}2} \\
 & & \wedge \diamondsuit\mathsf{gov} \circ \mathsf{bar{:}2} \\
\mathsf{LA}^2 & = & \diamondsuit\mathsf{gov} \bullet \diamondsuit\mathsf{opy} \bullet \mathsf{bar{:}2} \\
 & & \wedge \diamondsuit\mathsf{gov} \bullet \diamondsuit\mathsf{opy} \circ \mathsf{bar{:}2} \\
 & & \wedge \diamondsuit\mathsf{gov} \circ \diamondsuit\mathsf{opy} \bullet \mathsf{bar{:}2} \\
 & & \wedge \diamondsuit\mathsf{gov} \circ \diamondsuit\mathsf{opy} \circ \mathsf{bar{:}2} \\
\mathsf{LA}^3 & = & \diamondsuit\mathsf{gov} \bullet \diamondsuit\mathsf{opy} \bullet \mathsf{bar{:}2} \bullet \boxminus - \mathsf{gov} \\
 & & \wedge \diamondsuit\mathsf{gov} \bullet \diamondsuit\mathsf{opy} \circ \mathsf{bar{:}2} \bullet \boxminus - \mathsf{gov} \\
 & & \wedge \diamondsuit\mathsf{gov} \circ \diamondsuit\mathsf{opy} \bullet \mathsf{bar{:}2} \bullet \boxminus - \mathsf{gov} \\
 & & \wedge \diamondsuit\mathsf{gov} \circ \diamondsuit\mathsf{opy} \circ \mathsf{bar{:}2} \bullet \boxminus - \mathsf{gov}
\end{array} \tag{4.13}$$

(Observe that $\bullet$ binds stronger than $\circ$.) We shall show that if x is governed, the domains are as intended. For a proof consider a point x of a labelled tree $\mathbb{T}$. Let g denote the smallest node dominating both x and its governor (in fact the mother of both) and let m be the smallest maximal projection of g. Then $x < g \leq m$. So two cases arise, namely $g = m$ and $g < m$. In each cases LA^1 picks the right node. Likewise, if o denotes the smallest element containing x and a opacity element that IDC-commands x, then $x < o$. Three cases are conceivable, $o < g$, $o = g$ and $o > g$. However, if government can take place only under sisterhood, $o < g$ cannot occur. So $x < g \leq o \leq m$. For each of the four cases LA^2 picks the right node. Finally, for LA^3 there is an extra condition on m that it be ungoverned.

Notice that our translation is faithful to Koster's definitions only if the domains defined in (Koster, 1986) are monotone. This is by no means trivial. Namely, it is conceivable that a node has an ungoverned element y locally accessible^2, while the highest locally accessible^2 node, z, is governed. In that case (ignoring the opacity element for a moment) the domain of local accessibility^3 of y is z while the domain of x is strictly larger. We find no answer to this puzzle in the book because the domains are defined only for governed elements. But it seems certain that the monotone definition given here is the intended one.

It should be stressed that gov and opy are not specific labels. Their value may change from language to language, and from one type od dependency to another. Consequently, the local accessibility relations are parametrized with respect to the choice of particular governors and particular opacity elements. As an example, recall the Icelandic case again, where certain anaphors whose domain of accessibility^2 (typically the clause) can be extended in case the opacity element is *subjunctive*. Following our reduction, the domain of local accessibility^3 is defined by the first maximal projection that is not subjunctive, hence indicative. We take a primitive label ind to stand for *is indicative*. We write $\Diamond_{\leftarrow}\mathsf{x}$ to denote the set of nodes whose left sister is a x. Assume that government is to the right. Then for Icelandic we have the following special domain

$$
\begin{array}{rcl}
\mathsf{LA}^3 & = & \Diamond_{\uparrow}\mathsf{gov} \bullet \Diamond_{\uparrow}\mathsf{ind} \bullet \mathsf{bar:2} \bullet -\Diamond_{\leftarrow}\mathsf{gov} \\
 & & \wedge \Diamond_{\uparrow}\mathsf{gov} \bullet \Diamond_{\uparrow}\mathsf{ind} \circ \mathsf{bar:2} \bullet -\Diamond_{\leftarrow}\mathsf{gov} \\
 & & \wedge \Diamond_{\uparrow}\mathsf{gov} \circ \Diamond_{\uparrow}\mathsf{ind} \bullet \mathsf{bar:2} \bullet -\Diamond_{\leftarrow}\mathsf{gov} \\
 & & \wedge \Diamond_{\uparrow}\mathsf{gov} \circ \Diamond_{\uparrow}\mathsf{ind} \circ \mathsf{bar:2} \bullet -\Diamond_{\leftarrow}\mathsf{gov}
\end{array}
\tag{4.14}
$$

We notice in passing that recent results have put this analysis into doubt (see (Koster and Reuland, 1991)) but this is a problem of Koster's original definitions, not of this translation. What is a problem, however, is the standard opacity factor of an *accessible subject*. There is a rather technical notion in GB, that of a *SUBJECT* characterizing what can be the subject of an item as concerns binding theory. We will not go into that. While such notions can be easily handled introducing another boolean label, the accessibility condition presents real difficulties because to define in the system the distribution of the label corresponding to *SUBJECT* is a tricky affair. We will omit that here.

4.5 Nearness Defined by Path Sets

Command relations in an abstract form have first been introduced in (Barker and Pullum, 1990). Despite other constructs to be introduced later (such as nearness terms) command relations are rather stable under certain changes in the underlying grammar or transformations, which make them particularly useful for stratification of GB. Moreover, command relations have established themselves as a kind of lingua franca in linguistics. The problem is, however, that the interpretation of primitive command relations such as c-command is constantly changing, so that there is a certain need for standardization.

We will focus now on the definition of command relations via their domains, as outlined in (Barker and Pullum, 1990). Suppose that $R \subseteq T^2$ is a binary relation on $\mathbb{T}$. For two relations R, S we have $R = S$ exactly if for all nodes x we have $xR = xS$. (Barker and Pullum, 1990) list a number of desiderata a relation must meet in order to qualify for a command relation.

[Boundedness]	$rR = T$
[Constituency]	$(\forall x \exists y)(xR = \downarrow y)$
[Descent]	$(\forall xyz)(y \in xR \wedge z \leq y \rightarrow z \in xR)$
[Embeddability]	Command relations on a subtree are insensitive to material external to that subtree.
[Ambidextrousness]	Command relations are insensitive to the linear structure of the tree.

Note that Descent already follows from Constituency, which itself is a rather strong condition. In fact, Constituency can in this context be replaced by weak Constituency.

[Weak Constituency] If $y \in xR$ and $z \in xR$ then $w \in xR$ where w is the least node dominating both y and z.

Only seemingly weaker is an alternative definition where w is not required to be the minimal node dominating both x and y. To see that Weak Constituency and Descent imply Constituency on a finite tree observe that from Descent we can conclude that xR is the union of finitely many lower cones, namely $xR = \bigcup \langle \downarrow y : y \in xR \rangle$. But if $xR = \bigcup \langle \downarrow y_i : i \leq n \rangle$ then by Weak Constituency there is a w such that $\downarrow y_{n-1}, \downarrow y_n \subseteq \downarrow w \subseteq xR$ so that $\downarrow y_{n-1}$ and $\downarrow y_n$ can be replaced in this union by a single cone. Induction on n yields $xR = \downarrow w$ for some w.

So much for the independence of the postulates. The list of desiderata calls for some comments. It is not clear, for example, what is meant by Ambidextrousness if we consider a relation on some fixed ordered tree. Rather, these restrictions only make sense if we see them not as defining properties of relations over ordered trees but rather as defining properties of systems of relations over families of trees. So, given a set L of labels and a family (= set) $\mathfrak{F}$ of L-trees, we call $\mathfrak{R}$ an **abstract relation** or a **system of relations** over $\mathfrak{F}$ if $\mathfrak{R}$ is a function assigning to each tree $\mathbb{T}$ a binary relation $\mathfrak{R}(\mathbb{T})$. Then Ambidextrousness can be understood as saying that if $\sqsubset$ and $\sqsubset'$ are two different orderings on the same (labelled) tree $\mathbb{T}$ then with $\mathbb{T}_1 = \langle \mathbb{T}, \sqsubset \rangle$ and $\mathbb{T}_2 = \langle \mathbb{T}, \sqsubset' \rangle$ we have $\mathfrak{R}(\mathbb{T}_1) = \mathfrak{R}(\mathbb{T}_2)$. This property of ambidextrousness allows to treat command relations (i. e. abstract ones) as defined not over ordered trees but over unordered trees, by which we have reduced the complexity of the structures involved somewhat. The next great simplification results from Embeddability. However, this condition is somewhat vague in the sense that it is not clear what counts as a subtree. We present three consecutive definitions of subtrees which will yield different consequences of Embeddability. The strictest notion is that of a *constituent*. If at least all constituents are subtrees then Embeddability implies

[Well-Inclusion] $(\forall y)(yR \supseteq \downarrow y)$

For a proof let $R = \Re(\mathbb{T})$ and $S = \Re(\downarrow y)$ for some $y \in T$. Now let $z \leq y$. Then the domain of z in $\downarrow y$ should not be different in $\downarrow y$ if considered on its own than if considered as a subtree of $\mathbb{T}$. Hence, $zS = zR\cap \downarrow y$. Hence we have derived the principle that $\Re(\downarrow y) = \Re(\mathbb{T})\cap \downarrow y$ for any tree $\mathbb{T}$ and $y \in T$. Since y is the root of $\downarrow y$, we must have $\Re(\downarrow y)_y = \downarrow y$, whence $\Re(\mathbb{T})_y \supseteq \downarrow y$. Well-Inclusion and Constituency can be replaced by a single condition, namely

[Domain$^\uparrow$] $(\forall x)(\exists y \in \uparrow x)(xR = \downarrow y)$

And if a relation R satisfies Domain$^\uparrow$ in T, $R \cap (\downarrow y)^2$ satisfies Domain$^\uparrow$ on $\downarrow y$. For if $x \in \downarrow y$, let $xR = \downarrow w$ for some $w \in \uparrow x$. Then $(R \cap (\downarrow y)^2)_x = \downarrow w \cap \downarrow y = \downarrow \min\{y, w\}$. Since both $y, w \geq x$, $min\{y, w\} \geq x$. The characterization of command relations as proposed in (Barker and Pullum, 1990) therefore leads to the characterization of command relations as relations satisfying Domain$^\uparrow$. This includes the possibility to have $xR = \downarrow x$ for some or all x, which is excluded according to (Barker and Pullum, 1990). However, command relations satisfy a stronger condition, namely that the choice of the generator of the cone xR is not a member of the upper cone of x but rather an element properly dominating x if such an element exists, and r else. Let us define the **crown** of x to be the set ${}^\diamond x = \{r\} \cup \{y : y > x\}$. Then R is a command relation if and only if it satisfies

Domain$^\diamond$ $(\forall x)(\exists y \in {}^\diamond x)(xR = \downarrow y)$

The next class of subtrees derive from subsets U which are connected, have a root possibly different from r and leaves of $\mathbb{U} := \mathbb{T} \cap U$ must be leaves of $\mathbb{T}$. Such substructures are called **strong subtrees** here. Strong subtrees play an important role in Pesetzky's theory of extraction. This class contains all branches of $\mathbb{T}$. Hence, if $y \in T$ and $\mathbb{U}$ is a branch containing x then the relation $\Re(\mathbb{U})$ is nothing but $\Re(\mathbb{T}) \cap \mathbb{U}$. Since the domain of an element y is a constituent $\downarrow x$ with $x > y$, $x \in U$ and so the domains are determined from considering the branches alone.

The final, most weak definition is a subtree as a convex subset with maximal element. We call such subtrees **weak subtrees**. Positions are weak subtrees and by similar arguments the Embeddability constraint leads to a reduction to positions.

[Look Upwards] The domain of a node in a tree depends only on its position.

In addition to these requirements defined in (Barker and Pullum, 1990) for command relations in general there also is a restricted class of *fair* relations as defined in (Barker and Pullum, 1990). Fairness is easier to describe in terms of the generating relations than intrinsically. But there is a property of command relations that we have called *tightness* and that can characterize them. The idea is best explained in terms of movement. Suppose that x moves somewhere within its domain. Then the landing site of x cannot be the generating element of the downward cone xR, at least if movement is assumed to work as outlined in Chapter 3. And then tightness says that x commands no more elements in the new position than in the old one. Or, to rephrase that, tight command relations are such that no element can escape its domain via movement. Tightness will be shown to be strictly stronger than *monotonicity*, which is the condition that $x \leq y$ implies $xR \subseteq yR$; this seems to be universally satisfied by all command relations in language. It seems therefore that monotony is an essential property of command relations; but from a formal point of view we would like not to impose it as a condition sine qua non.

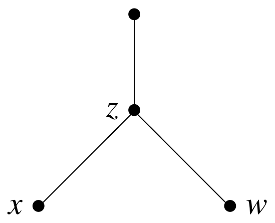
[Monotony] $(\forall xy)(x \leq y. \rightarrow .xR \subseteq yR)$

[Tightness] $(\forall xy)(y \in xR \wedge (\downarrow y \neq xR). \rightarrow .yR \subseteq xR)$

That Tightness implies Monotony is seen thus. Suppose that $x \leq y$. Then either $x = y$ in which case $xR \subseteq yR$ is trivially true; or $x < y$ and then $\downarrow x \subsetneq \downarrow y \subseteq yR$ and hence $\downarrow x \neq yR$. By Tightness $xR \subseteq yR$, as required. It is to be added here that fair relations cannot be characterized by their behaviour on single branches unlike monotone relations which just need to be monotone on each branch. Tightness restricted to branches is the condition

$$(\forall xyz)(x < y < z \wedge \downarrow z = xR. \rightarrow .yR = \downarrow z)$$

This property is not identical to tightness. Consequently, tightness is a property of relations *over the whole tree*. The following tree with $wR = rR = \downarrow r$ and $xR = \downarrow z$ shows that a relation which is tight on all branches need not be tight on the tree.

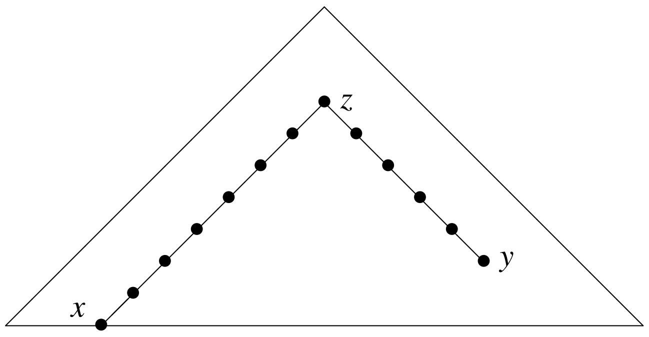


(4.25)

The concept of nearness is based on a notion of distance in trees. This notion is not straightforward because trees are not linear and so do not have a direct distance. However, if x and y are points in a tree, then there always exists a z such that $z \geq x$ and $z \geq y$. Moreover, there is a least such z because $\uparrow x$ and $\uparrow y$ are linear. The distance between x and y is then a pair of distances, one from x to z and the other from y to z.

Definition 4.5.1 *Let $x, y \in T$. We call $z := \min(\uparrow x \cap \uparrow y)$ the* ***crosspoint*** *of x and y. The* ***distance*** *between x and y is given by a pair $\langle \lambda, \rho \rangle$ of natural numbers defined by $\lambda := \sharp(\uparrow x \cap \downarrow z) - 1$ and $\rho := \sharp(\uparrow y \cap \downarrow z) - 1$. We write $dist(x, y) := \langle \lambda, \rho \rangle$. We also say* *that x and y are $\langle\lambda,\rho\rangle$-**near** if there exists a $u \geq x, y$ such that $\lambda = \sharp(\uparrow x \cap \downarrow u) - 1$ and $\rho = \sharp(\uparrow y \cap \downarrow u) - 1$. Obviously, x and y are always dist(x,y)-near.*

(4.26)



In the picture we have $dist(x,y) = \langle 7,5\rangle$ since $\sharp(\uparrow x \cap \downarrow z) = 8$ and $\sharp(\uparrow y \cap \downarrow z) = 6$. This calls for some comments. First, let us check this concept for the simple case that $y \geq x$. Then $y = z$ and so $\rho = 0$, because $\uparrow y \cap \downarrow y = \{y\}$. Moreover, $\lambda = \sharp(\uparrow x \cap \downarrow y) - 1$ measures how many nodes one has to travel before reaching y. If y is identical to x then $\lambda = 0$ and if y covers x then $\lambda = 1$; and so forth. Thus we have $x = y$ iff $dist(x,y) = \langle 0,0\rangle$, $x \leq y$ iff $dist(x,y) = \langle\lambda,0\rangle$ and $x \geq y$ iff $dist(x,y) = \langle 0,\rho\rangle$. Notice that while normally distances are symmetric, this is not so here, by definition of distance as a pair. We might have defined the distance as a set $\{\lambda,\rho\}$ in which case we would have had symmetry, but it is not our aim here to have full congruence with the classical definition.

In fact, we want to take advantage of the asymmetry of the definition in a special way. Notice, namely, that if y has distance $\langle 1,0\rangle$ from x it covers x while x covers y if the distance from x to y is $\langle 0,1\rangle$. We are interested in keeping a distinction between the up- and the down-going direction of the two components. To this end defined the following functions

(4.27)
$$\begin{array}{lcl}\mathsf{next}(S) & := & \{y : (\exists x \in S)(x \prec y)\}\\ \mathsf{prev}(S) & := & \{y : (\exists x \in S)(y \prec y)\}\end{array}$$

These functions are defined on subsets of T. $\mathsf{next}(S)$ returns all covers for nodes of S while $\mathsf{prev}(S)$ returns all nodes covered by nodes form S. Notice that $\mathsf{next}(S)$ can be defined as $\mathsf{next}(S) := \{\mathsf{next}(x) : x \in S\}$ where $\mathsf{next}(x)$ is the following partial function.

(4.28) $$\mathsf{next}(x) := \begin{cases} y & \text{if } y \text{ covers } x\\ \text{undefined} & \text{else}\end{cases}$$

For $\mathsf{prev}(S)$ a similiar reduction is not possible. We define iterations of these functions as usual; $\mathsf{next}^0(S) := S$, $\mathsf{next}^{(n+1)}(S) := \mathsf{next}(\mathsf{next}^n(S))$ $\mathsf{prev}^0(S) := S$, $\mathsf{prev}^{(n+1)}(S) := \mathsf{prev}(\mathsf{prev}^n(S))$. These iterations will be used to define the symbols $\langle n|$ and $|n\rangle$. For a set S let $S\langle n| = \mathsf{next}^n(S)$ and $S|n\rangle := \mathsf{prev}^n(S)$. Moreover, $S\langle m|n\rangle = \mathsf{prev}^n(\mathsf{next}^m(S))$.

(The notation is Dirac's notation from quantum mechanics; there the idea is that there are two types of vectors, one from the space and one from the dual. One is denoted by $\langle v|$ and the other by $|v\rangle$. The skalar product of two vectors v, w is $\langle v|w\rangle$.) The same number thus represents both an up- and an downgoing function, depending on the context. Notice that these functions can be iterated arbitarily. For example, $S\langle n|m\rangle\langle \ell|$ denotes a set in T. However, the following rules are directly verified:

$$\langle m|\langle n| = \langle m+n|, |m\rangle|n\rangle = |m+n\rangle,$$

$$|m\rangle\langle n| \subseteq |m-n\rangle \text{ if } m \geq n,$$

$$|m\rangle\langle n| \subseteq \langle n-m| \text{ if } m \leq n$$

We write $x|m\rangle$, $x\langle m|$ for $x|m\rangle$ and $x\langle m|$. Note that $|m\rangle\langle n|$ need not be equal to $|m-n\rangle$. For example, if y is a preleaf and $m = 2, n = 1$ then $y|2\rangle = \emptyset$, whence $y|2\rangle\langle 1| = \emptyset$. But of course $y|1\rangle \neq \emptyset$. Notice that there are no laws reducing $\langle n|m\rangle$. Notice that y is $\langle \lambda, \rho\rangle$-near to x iff $y \in x\langle \lambda|\rho\rangle$.

4.6 Structured Distance

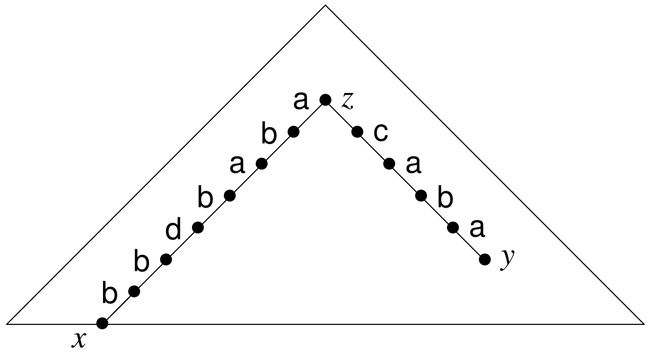
The definition of distance is now carried over to labelled trees. Of course, the numerical distance defined earlier remains a valid definition but the labellings allows for a more subtle definition of distance that will be called *structured.* Taking two nodes x and y and the crosspoint z the structures $\uparrow x \cap \downarrow z$ and $\uparrow y \cap \downarrow z$ are now linear labelled trees. As for such trees, there is a direct correspondence between the labelling and the *string* of labels $\ell(x)$ when the nodes are ordered by $<$, i. e. read from bottom to top. Namely, define for a linear, labelled tree $\mathfrak{T} = \langle \mathbb{T}, \sqsubset, \ell\rangle$ where $\mathbb{T} = \{0, 1, \ldots, n\}$ ordered by $<$

$$\mathbf{tp}(\mathfrak{T}) = \langle \ell(y_i) : 1 \leq i \leq n\rangle$$

Two such strings describe the voyage from x to z within an arbitrary tree $\mathfrak{T}$; the type of the weak subtree from x to z and the type of the weak subtree from y to z.

Definition 4.6.1 *Let $\mathfrak{T}$ be a labelled (ordered) tree, $x, y \in T$ with crosspoint z. The **structured distance** from x to y is the pair*

$$dist(x, y) = \langle \mathbf{tp}(\uparrow x \cap \downarrow z), \mathbf{tp}(\uparrow y \cap \downarrow z)\rangle$$



In the tree shown above we have $dist(x, y) = \langle \mathsf{bbdbaba}, \mathsf{abaca}\rangle$. Again, this definition is not symmetric, e. g. $dist(y, x) = \langle \mathsf{abaca}, \mathsf{bbdbaba}\rangle$. We have $x \leq y$ iff $dist(x, y) = \langle \lambda, \epsilon\rangle$, $x \geq y$ iff $dist(x, y) = \langle \epsilon, \rho\rangle$ and, consequently, $x = y$ iff $dist(x, y) = \langle \epsilon, \epsilon\rangle$. As before, the two strings λ and ρ occur with different interpretation. The functions $\langle -|$ and $|-\rangle$ are defined inductively as follows.

$$S\langle \mathsf{x}| = \{y : (\exists x \in S)(x \prec y \text{ and } \ell(y) \sqsubseteq \mathsf{x}\}$$

$$S|\mathsf{x}\rangle = \{y : (\exists x \in S)(y \prec x \text{ and } \ell(x) \sqsubseteq \mathsf{x}\}$$

$$S\langle \mathfrak{s} \cdot \mathfrak{t}| = S\langle \mathfrak{s}|\langle \mathfrak{t}|$$

$$S|\mathfrak{s} \cdot \mathfrak{t}\rangle = S|\mathfrak{s}\rangle|\mathfrak{t}\rangle$$

Here $\mathfrak{s}$, $\mathfrak{t}$ are sequences of labels and $\mathfrak{s} \cdot \mathfrak{t}$ their concatenation. We write $x\langle \mathfrak{s}|$ for $\{x\}\langle \mathfrak{s}|$ and also $x|\mathfrak{t}\rangle$ for $\{x\}|\mathfrak{t}\rangle$. Since from $y, y' \in x\langle \mathfrak{s}|$ follows $y = y'$, we often write $y = x\langle \mathfrak{s}|$. Furthermore, we note that if in a tree the isomorphism type of the position $\uparrow y$ determines y uniquely, then for all $y, y' \in x|\mathfrak{t}\rangle$ y and y' are sisters, not necessarily identical. Nevertheless, in this case we also write $y = x|\mathfrak{t}\rangle$ to denote any of y and its sisters. Notice that $x\langle \mathfrak{s}|\mathfrak{t}\rangle$ is empty if the last symbols of $\mathfrak{s}$ does not match the first symbol of $\mathfrak{t}$, i. e. if $\mathfrak{s} = \sigma_1 \cdot \ldots \cdot \sigma_m$ and $\mathfrak{t} = \tau_1 \cdot \ldots \cdot \tau_n$ and $\tau_1 \neq \sigma_n$. If $S\langle \mathfrak{s}|$ (or $S|\mathfrak{t}\rangle$) is empty we also say that $\langle \mathfrak{s}|$ ($|\mathfrak{s}\rangle$) is *undefined* on S. Furthermore, the following universally true.

1. $x\langle \epsilon| = x, x|\epsilon\rangle = x$.
2. $x|\mathfrak{s}\rangle\langle \mathfrak{s}^T| = x$ if $x|\mathfrak{s}\rangle$ is defined.
3. $x|\mathsf{a}\rangle$ is defined iff $\ell(x) \sqsubseteq \mathsf{a}$ and $\exists y : x \prec y$.
4. $x\langle \mathsf{a}|$ is defined iff $\exists y : y \prec x$ and $\ell(y) \sqsubseteq \mathsf{a}$.

This definition is extended to sets of strings. Given a set V of labels and $L, R \subseteq V^*$ then let

$$S\langle L| = \bigcup \langle S\langle \mathfrak{l}| | \mathfrak{l} \in L\rangle$$

$$S|R\rangle = \bigcup \langle S|\mathfrak{r}\rangle | \mathfrak{r} \in R\rangle$$

Proposition 4.6.2 (Red Contraction) $x|\mathfrak{s}\rangle\langle\mathfrak{t}|$ *is defined only if either* $\mathfrak{s}^T$ *is a prefix of* $\mathfrak{t}$ *or* $\mathfrak{t}^T$ *is a suffix of* $\mathfrak{s}$*. In the first case* $\mathfrak{t} = \mathfrak{s}^T \cdot \mathfrak{u}$ *and so* $x|\mathfrak{s}\rangle\langle\mathfrak{t}| = x\langle\mathfrak{u}|$*; in the second case* $\mathfrak{s} = \mathfrak{u} \cdot \mathfrak{t}^T$ *and* $x|\mathfrak{s}\rangle\langle\mathfrak{t}| = x|\mathfrak{u}\rangle$.

Proof. Straightforward verification. ⊣

Notice that in this contraction law we do not have *iff*, but only *only if.* This is why it is called *red*. For example, the polynomial $\langle \mathsf{a}\cdot\mathsf{b}|\mathsf{b}\cdot\mathsf{b}\rangle\langle\mathsf{b}\cdot\mathsf{b}\cdot\mathsf{c}|$, if defined is equal to $\langle\mathsf{a}\cdot\mathsf{b}\cdot\mathsf{c}|$. Yet, the latter is defined on certain trees, where the former is not, and so equality does not always hold. However, the following can be verified as easily.

Proposition 4.6.3 (Green Contraction) *For all* $\mathfrak{s}$

$$|\mathfrak{s}\rangle\langle\mathfrak{s}^T|\mathfrak{s}\rangle = |\mathfrak{s}\rangle, \langle\mathfrak{s}|\mathfrak{s}^T\rangle\langle\mathfrak{s}| = \langle\mathfrak{s}|$$

Proof. If $|\mathfrak{s}\rangle$ is defined at x, $|\mathfrak{s}\rangle\langle\mathfrak{s}^T|\mathfrak{s}\rangle$ is defined at x as well, and they are equal because if $y \in x|\mathfrak{s}\rangle$ then $x \in y\langle\mathfrak{s}^T|$, whence the latter is defined. By Red Contraction, $y\langle\mathfrak{s}^T|\mathfrak{s}\rangle = y$ and this shows equality. Similarly for the other case. ⊣

Here, equality strictly holds, so this law can be used at any time. From now on we call $\langle L|$ and $|M\rangle$ **paths** and the sets L, M the **path sets**. From paths we can build up polynomials in the following way. A **nearness polynomial** p is a function composed from $\langle L|$ and $|L\rangle$ for $L \subseteq V^*$ using function composition and union. We define the union by $S(p \cup q) = Sp. \cup .Sq$. We call p **spherical** if it is composed without union. Union distributes over composition and so any nearness polynomial is a union of spherical polynomials. The nearness polynomials form a structure $\langle \mathfrak{P}, \cdot, \cup\rangle$ where $\langle\mathfrak{P}, \cdot\rangle$ is a semigroup and $\langle\mathfrak{P}, \cup\rangle$ is a semi-lattice and $\cdot$ distributes over $\cup$. This implies among other that there is an ordering $\subseteq$ of nearness polynomials. We have $p \subseteq q$ iff for all trees and all nodes x, y if $y \in xp$ then $y \in xq$. Let $p = q_1 \cup \ldots q_m$ where q_i are spherical. The degree of the spherical polynomial $dg(q_i)$ is the number of simple $\langle L|$ and $|L\rangle$ composed, e. g. $dg(\langle L|\langle M|N\rangle) = 3$. We then define the **degree** of p as $dg(p) = max\{dg(q_i) : 1 \leq i \leq m\}$. We can show that under some conditions on the path sets any nearness polynomial can be reduced to a polynomial of degree at most 2. Call $\langle L|$, $|R\rangle$ **s-closed** if it is closed under suffixes. Call it **p-closed** if it is closed under prefixes. p is **sp-closed** if all $\langle L|$ paths are s-closed and all $|R\rangle$ paths are p-closed. The set Sp is called a **n-sphere around** S if p is a spherical polynomial of degree n, otherwise it is called a **generalized n-sphere**. If p is sp-closed, a generalized sphere around S is a convex subset containing S. The proof of this reduction lies in a generalization of Contraction to sets. Let L, M be sets of strings. Then write $L^{-1} \cdot M = \{m : (\exists \ell \in L)(\ell \cdot m \in M)\}$ and $L \cdot M^{-1} = \{\ell : (\exists m \in M)(\ell \cdot m \in L)\}$.

Lemma 4.6.4 (Set Contraction) $|L\rangle\langle M| \subseteq |L(M^T)^{-1}\rangle \cup \langle (L^T)^{-1}M|$.

Proof. $L(M^T)^{-1}$ is the language where all strings are computed in which a postfix of a $\ell \in L$ has been cancelled by a $m^T \in M^T$. Dually, $(L^T)^{-1}M$ is the language where all strings are computed where a prefix of an $m \in M$ has been neutralized by a $\ell^T \in L^T$. Now to conclude that the two polynomials are equal, is nevertheless a mistake. Rather, if a polynomial on the left hand side is defined, it is by Red Contraction equal to a polynomial on the right hand side. ⊣

Lemma 4.6.5 *Let p be sp-closed. Then there exists a sp-closed nearness polynomial q of degree at most 2 such that for all trees and all sets S: $S\,p = S\,q$.*

Proof. Obviously, we can reduce $\langle L|\langle M|$ to $\langle L\cdot M|$ and $|L\rangle|M\rangle$ to $|L\cdot M\rangle$. The interest now lies in possible reductions of $\langle L|M\rangle$ and $|L\rangle\langle M|$. The first is not reducible in the general case; let us concentrate on the second. Here we have the following inequalities

$$|L(M^T)^{-1}\rangle \cup \langle (L^T)^{-1}M| \subseteq |L\rangle\langle M| \subseteq |L(M^T)^{-1}\rangle \cup \langle (L^T)^{-1}M|$$

The first holds by sp-closure, the second by Set Contraction. Notice namely that by sp-closure, $\epsilon \in L$ and $\epsilon \in M$, hence $\epsilon \in L^T, M^T$ and finally $L(M^T)^{-1} \subseteq L$ as well as $M(L^T)^{-1} \subseteq M$. So

$$|L(M^T)^{-1}\rangle \subseteq |L\rangle \subseteq |L\rangle\langle\epsilon| \subseteq |L\rangle\langle M|$$
$$\langle (L^T)^{-1}M| \subseteq \langle M| \subseteq |\epsilon\rangle\langle M| \subseteq |L\rangle\langle M|$$

Thus

$$|L\rangle\langle M| = |L(M^T)^{-1}\rangle \cup \langle (L^T)^{-1}M|$$

Any 2-sphere $|-\rangle\langle-|$ is thus reducible to a union of 1-spheres. Hence, any n-sphere for $n \geq 3$ is reducible to a union of 2-spheres. ⊣

The previous lemma shows that there is a a special class of polynomials to which we can reduce all sp-closed polynomials.

Definition 4.6.6 *An* **oval nearness polynomial** *is a 2-sphere $\langle L|R\rangle$ where $L, R \subseteq V^*$. If S is a set in a tree and p an oval nearness polynomial then $S\,p$ is called an* **oval around** *S.*

Theorem 4.6.7 *Let p be a sp-closed nearness polynomial. Then p is equivalent to a union of oval nearness polynomials.*

Proof. By Lemma 4.6.5 any sp-closed polynomial is equivalent to a sp-closed polynomial of degree 2. Moreover, 2-spheres of type $|-\rangle\langle-|$ (i. e. non oval 2-spheres) can be reduced to a union of 1-spheres. Now notice that $\langle L| = \langle L|\epsilon\rangle$ and $|M\rangle = \langle\epsilon|M\rangle$, so 1-spheres are ovals. ⊣

Suppose that a tree is such that for any x, all daughters have different label if non-terminal. Then from any node x any other non-terminal y can be adressed by the distance $dist(x, y) = \langle\mathfrak{s}, \mathfrak{t}\rangle$ plus the label of y. However, the set $x\langle\mathfrak{s}|\mathfrak{t}^T\rangle$ not only contains y but also all of its sisters. If we want to define the set $\{y\}$ alone, we have to use the following trick. Let $\ell(y) = \mathsf{a}$. Then $\{y\} = x\langle\mathfrak{s}|\mathfrak{t}^T \cdot \mathsf{a}\rangle\langle\mathsf{a}|$. Hence, to define certain spheres, we really need polynomials of degree 3.

Proposition 4.6.8 *Suppose $\mathfrak{T}$ is a tree such that all nodes are uniquely identified by their position. Moreover, let a nonterminal x either have a single, terminal daughter or let otherwise all daughters be nonterminal. Then let $x \in T$ be any point and Y be any subset of T: Y is a generalized sphere around x.*

Proof. It suffices to show that $\{y\}$ is a sphere around x for all $\{y\}$. Now, if y is non-terminal, we have shown this already. If, however, y is terminal and $dist(x, y) = \langle\mathfrak{s}, \mathfrak{t}\rangle$, then $x\langle\mathfrak{s}|\mathfrak{t}^T\rangle = \{y\}$. ⊣

In what is to follow we will call attention to the fact that the sets L in the nearness polynomials $\langle L|$, $|L\rangle$ are by definition languages over the vocabulary of non-terminal symbols. This allows to study nearness terms with the tools of formal language theory. We say that a nearness polynomial p is *recursive*, *context-free*, *regular* iff the occurring path sets are *recursive*, *context-free*, *regular* as languages over V. Regular nearness terms play a key role in our investigation. It will be proved that definable command relations can also be seen as regular nearness terms. Moreover, regular nearness terms are well-behaved from a syntactic point of view.

4.7 The Embeddability Constraint

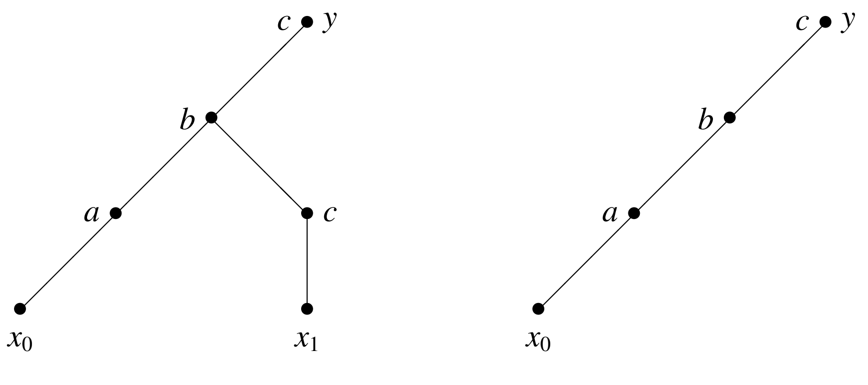
The purpose of this section is to prove that the Embeddability formulated earlier leads to a similar contraction of polynomials than does sp-closure. Thus, if there is a grammatical law making use of an abstract relation we can assume this relation to be a union of ovals if it satisfies Embeddability. The techniques are not difficult but rather delicate in that one has to beware of certain exceptional cases.

We begin with a simple sphere $\mathbb{S} = \langle\mathfrak{u}_1|\mathfrak{d}_1\rangle\langle\mathfrak{u}_2|\mathfrak{d}_2\rangle \ldots \langle\mathfrak{u}_n|\mathfrak{d}_n\rangle$. If we allow empty strings to occur, this is the most general type of a sphere. There is a sense in which this sphere decribes a *voyage* on a tree, which is a bit tricky to define. First, the paths $\langle\mathfrak{u}_i|$, $|\mathfrak{d}_i\rangle$ can be decomposed into elementary steps $\langle\sigma_i^j|$, $|\tau_i^j\rangle$ where σ_i^j and τ_i^j are non-terminal symbols. The number of elementary steps shall be n. Intuitively, an $\mathbb{S}$-voyage, that is, a voyage defined by the sphere $\mathbb{S}$, is a partial function $v : n \times T \to T$ conforming to the description given by the elementary steps. Let $\mathbb{S} = f_1 \cdot f_2 \cdot \ldots \cdot f_n$ the decomposition of $\mathbb{S}$ into elementary functions. Choose a point x and put $v(0) := x$. Inductively, $v(i+1) := v(i)f_i$. $v(i+1)$ is undefined if either $v(i)$ is undefined or f_i is undefined on $v(i)$. The function v is also called the $\mathbb{S}$*-voyage of* x. If v is a total function, the voyage is said to be *complete*. The **trace** of a voyage v is simply $im[v] = \{v(i) | 0 \leq i \leq n\}$. $v(0)$ is the **begin**, $v(n)$ the **end** of the voyage. In a tree, $y \in x\mathbb{S}$ iff there is a $\mathbb{S}$-voyage with begin x and end y.

Lemma 4.7.1 *Let* $\mathbb{S} = \langle \mathfrak{u}_1 | \mathfrak{d}_1 \rangle \langle \mathfrak{u}_2 | \mathfrak{d}_2 \rangle \ldots \langle \mathfrak{u}_n | \mathfrak{d}_n \rangle$ *be an n-sphere and* $\mathfrak{T}$ *a tree. The trace of an* $\mathbb{S}$*-voyage in* $\mathfrak{T}$ *is a weak subtree.*

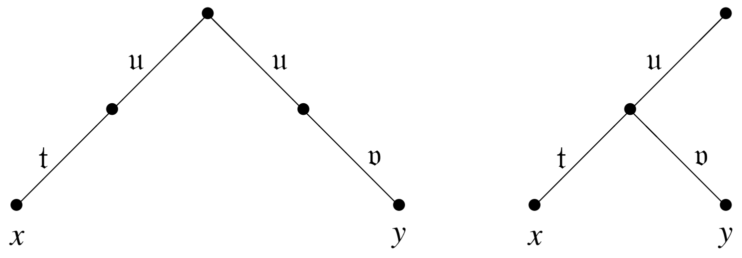
Proof. Notice that by definition, the trace of a voyage is a connected subset of $\mathfrak{T}$. Connected subsets are weak subtrees. Namely, let $U = \{x_1, \ldots, x_n\} \subseteq T$ be a connected subset. We construct a sequence of z_i such that $z_i \geq x_j$ for all $j \leq i$. Evidently, $z_1 = x_1$ is a good start. Then $(\uparrow x_1 \cup \uparrow x_2) \cap \downarrow z_2 \subseteq U$ for the crosspoint z_2, by connectedness. Moreover, $z_2 \in U$. Now take x_3 and let z_3 be the crosspoint of x_3 and z_2. Again, $z_3 \in U$. Moreover, $z_3 \geq x_1, x_2, x_3$. And so forth. $z_n \in U$ and $z_n \geq x_i$ for all i. Hence z_n is the root of $\mathfrak{U}$ in $\mathfrak{T}$. ⊣

Voyages are certainly not always the best way to go from one point to another. It is clear that if x, y are points and $dist(x, y) = \langle \mathfrak{u}, \mathfrak{d} \rangle$ then $\langle \mathfrak{u} | \mathfrak{d} \rangle$ is the shortest possible voyage from x to y. Moreover, a voyage v is shortest if it is injective as a function. We will show in the subsequent investigation that EMBEDDABILITY is tantamount to requiring that if p is a polynomial and $y \in xp$, then indeed the $\widehat{dist(x, y)}$-voyage is a voyage for some $\mathbb{S} \subseteq p$. But first an example. Let $\mathbb{S} = \langle \mathsf{ab} | \mathsf{bc} \rangle \langle \mathsf{cbc} |$, $\widehat{\mathbb{S}} = \langle \mathsf{abc} |$. $\widehat{\mathbb{S}}$ results from $\mathbb{S}$ by Red Contraction.

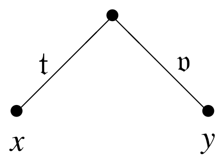


There is exactly one $\mathbb{S}$-voyage from x_0 to y in the first tree, but there is none in the second. The attempt fails because $\mathbb{S}$ also contains what amounts to a detour via x_1. Red Contraction cuts short such detours with the results that $\widehat{\mathbb{S}}$ is now well defined on the second tree.

Of course, it is also defined on the larger tree. Notice also that the traces of voyages have the property that if they can be embedded into a tree $\mathfrak{T}$ then there is a voyage from the embedded begin to the embedded end. In a sense, traces correspond to minimal subtrees on which a voyage can be defined. However, as we have seen, a voyage may contain more than just the point needed to join the begin and end. Now let us see what Embeddability amounts to: if v is an $\mathbb{S}$-voyage for some $\mathbb{S} \subseteq p$ with begin x and end y, then there must be a $\widehat{\mathbb{S}} \subseteq p$ with begin x and end y which is defined on the points connecting x and y. Since the latter are just the ones on the $dist(x, y)$-voyage we restate this as follows. If $\mathbb{S} \subseteq p$ and $y \in xp$ and $dist(x, y) = \langle \mathfrak{u}, \mathfrak{d} \rangle$ then $\langle \mathfrak{u} | \mathfrak{d} \rangle \subseteq p$. We will restate this once more now using closure under Contraction. Namely, when we apply Red Contraction we can remove certain unnecessary detours leading us through extra points. Indeed, Red Contraction leads to a polynomial that is equal to the one given in case the two are defined. Moreover, if $\mathbb{S}$ is defined for x and $y \in x\mathbb{S}$ then the contractum $\widehat{\mathbb{S}}$ is defined on some proper subtree. So $\widehat{\mathbb{S}} \subseteq p$ for each $\mathbb{S} \subseteq p$. Notice that Green Contraction leads to a shorter voyage on no less points. It just concerns an up- and down-movement on the same set of points. There is, finally, a third type of Contraction that we need to consider. Suppose, namely, that we have the sphere $\mathbb{S} = \langle \mathfrak{t} \cdot \mathfrak{u} | \mathfrak{u}^T \cdot \mathfrak{v} \rangle$. Then there are two possible types of voyages that can be defined on trees.



While the first trace contains no superfluous points because it goes down a different path than the one it went up, the second contains a detour using the path $\mathfrak{u}$. The latter has a contractum $\langle \mathfrak{t} | \mathfrak{v} \rangle$ because it should be defined also on the minimal subtree connecting x and y — at least if Embeddability is required.



Let us call a sphere $\langle\mathfrak{u}|\mathfrak{u}^T\rangle$ a **yo-yo-sphere**. (Well, since our trees grow upside down, so is yo-yo-movement an up- and down-movement rather than a down- and up-movement.) p is closed under **yo-yo-contraction** if $q\langle\mathfrak{u}|\mathfrak{u}^T\rangle q' \subseteq p$ implies $q \cdot q' \subseteq p$.

Proposition 4.7.2 (Yo-Yo-Contraction) *If p satisfies* Embeddability *then p is closed under yo-yo-contraction.* ⊣

Lemma 4.7.3 *An oval p satisfies* Embeddability *iff it is closed under yo-yo-contraction.*

Proof. The direction from left to right is covered by Lemma 4.7.2. Consider a $\langle\mathfrak{u}|\mathfrak{d}\rangle \subseteq p$. Let $y \in x\langle\mathfrak{u}|\mathfrak{d}\rangle$ and $dist(x, y) = \langle\mathfrak{g}, \mathfrak{h}\rangle$. Then we must have $\mathfrak{u} = \mathfrak{g} \cdot \mathfrak{x}$ and $\mathfrak{d} = \mathfrak{x}^T \cdot \mathfrak{h}$. By closure under yo-yo-contraction, $\langle\mathfrak{g}|\mathfrak{h}\rangle \subseteq p$. But this means that p satisfies Embeddability. ⊣

Theorem 4.7.4 *An abstract relation $\mathfrak{R}$ satisfies* Embeddability *iff it is a (possibly infinite) union of ovals and closed under yoyo-contraction.*

Proof. Let $\mathfrak{R}$ be given. By Embeddability, $\mathfrak{R}$ is faithfully represented by

$$\mathfrak{R} = \bigcup \langle\langle\langle\mathfrak{g}|\mathfrak{h}\rangle|\langle\mathfrak{g}, \mathfrak{h}\rangle = dist(x, y) \text{ for some } \mathfrak{T} \text{ and } x, y \in T \text{ with } x\mathfrak{R}(\mathfrak{T})y\}$$

Notice, that this polynomial is indeed closed under yo-yo-contraction. This proves the direction from left to right. The other direction is easy. ⊣

This theorem shows that Embeddability reduces relations to unions of ovals. Interesting is what happens to *regular polynomials*. Assume that we have a regular nearness polynomial p. Then observe that Set Contraction if applied to regular spheres yields regular spheres. Namely, by Set Contraction $|L\rangle\langle R| \subseteq |L(R^T)^{-1}\rangle \cup \langle(L^T)^{-1}R|$ and if L, R are regular, so are $L(R^T)^{-1}$ and $(L^T)^{-1}R$. Moreover, if p satisfies Embeddability then $|L\rangle\langle R| \subseteq p$ implies $|L(R^T)^{-1}\rangle \cup \langle(L^T)^{-1}R| \subseteq p$. Thus in p we can replace $|L\rangle\langle R|$ by $|L(R^T)^{-1}\rangle \cup \langle(L^T)^{-1}R|$. We thus effectively reduce p to a union of regular ovals. The problem is now what happens to the yo-yo-closure. Here, we find ourselves in trouble because yo-yo-contraction has no set-equivalent. If $\langle L|R\rangle$ is an oval, then we would guess that the yo-yo-contracta are given by $\langle L(R^{PT})^{-1}|(L^{ST})^{-1}R\rangle$ but it is not hard to see that this overgenerates. Only if we assume that L and R are closed with respect to prefixes and suffixes, we stand a chance. If this is the case we say that $\langle L|R\rangle$ is **ps-closed**, dually to sp-closure. Notice namely, that if $\langle L|R\rangle$ is ps-closed and $\langle\mathfrak{g} \cdot \mathfrak{u}|\mathfrak{u}^T \cdot \mathfrak{h}\rangle \in \langle L|R\rangle$ then $\langle\mathfrak{g}|\mathfrak{h}\rangle \in \langle L|R\rangle$ by prefix closure of L and suffix closure of R.

Theorem 4.7.5 *Let p be a regular, ps-closed nearness polynomial. Then p satisfies* Embeddability *iff p is equivalent to a finite union of ps-closed regular ovals.* ⊣

In order to characterize command relations via nearness polynomials we can use these results the quite effectively. Command relations satisfy Embeddability and are therefore an infinite union of ovals and are closed under yo-yo-contraction. The next property we consider is Domain$^{\uparrow}$. Together with Embeddability it implies that if $\langle \mathfrak{u}|\mathfrak{d}\rangle \subseteq \mathfrak{R}$ then $\langle \mathfrak{u}| \in \mathfrak{R}$ as well, because if $dist(x,y) = \langle \mathfrak{u}, \mathfrak{d}\rangle$ then for the crosspoint z $dist(x,z) = \langle \mathfrak{u}, \epsilon\rangle$. By Embeddability, in the subtree $\uparrow x \cap \downarrow z$ $x\mathfrak{R}z$ as well. Now let $\mathfrak{v}$ be any string. There exists a tree $\mathfrak{U}$ such that $dist(x,y) = \langle \mathfrak{u}, \mathfrak{v}\rangle$. (Just glue a branch of type $\mathfrak{v}$ onto z.) By Domain$^{\uparrow}$, $x\mathfrak{R}(\mathfrak{U})y$. Thus, we have seen that if $\mathfrak{R}$ satisfies both Embeddability and Domain$^{\uparrow}$ it is a union of ovals of the form $\langle L|V^*\rangle$. But for all $L_i, i \in I$ and R

$$\bigcup_{i\in I}\langle L_i|R\rangle = \langle\bigcup_{i\in I} L_i|R\rangle$$

so that, finally, $\mathfrak{R}$ can be reduced to a single oval of the form $\langle L|V^*\rangle$. By yo-yo-closure, L must be closed under prefixes, as is easily checked.

Theorem 4.7.6 *An abstract relation satisfies* Embeddability *and* Domain$^{\uparrow}$ *iff it is an oval* $\langle L|V^*\rangle$ *with* $L = L^P$. ⊣

Let us distinguish between a *left* oval and a *right* oval; an oval is a **left** oval if it is of type $\langle L|V^*\rangle$ and a right oval if it is of type $\langle V^*|R\rangle$. We have found that command relations are special left ovals.

Theorem 4.7.7 *An* $\mathfrak{R}$ *is a command relation iff it is a left oval* $\langle L|V^*\rangle$ *with* $L \supseteq V$ *and* L *closed under prefixes. Furthermore,* $\mathfrak{R}$ *is monotone iff* L *is in addition closed under suffixes.*

Proof. From left to right we only need to observe that since for $x \prec y$ we have $y \in \mathfrak{R}(\mathfrak{T})_x$ regardless of the label of y, and since $dist(x,y) = \langle \sigma, \epsilon\rangle$ we must have $a \in V$ for every $a \in V$. This concludes the proof of the direction from left to right. For the other direction we need to show that $\langle L|V^*\rangle$ for a $L = L^P \supseteq V$ is a command relation. We know by the previous theorem that it satisfies Embeddability and Domain$^{\uparrow}$. Moreover, Domain$^{\diamond}$ is also quite clear. Ambidextrousness is satisfied because structured distance is completely independent of the linear ordering $\sqsubset$. Now for Monotonicity observe that it is equivalent to the condition that if $x \leq y \leq f_R(x)$ then $f_R(x) \in R_y$. So suppose that $f_R(x) = x\langle \mathfrak{r}|$ and that $x \leq y \leq f_R(x)$. Then $y = x\langle \mathfrak{g}|$ for a prefix $\mathfrak{g}$ of $\mathfrak{r}$. It follows that $\mathfrak{r} = \mathfrak{g} \cdot \mathfrak{h}$ with a suffix $\mathfrak{h}$ and $f_R(x) = x\langle \mathfrak{g} \cdot \mathfrak{h}| = y\langle \mathfrak{h}|$. Since $f_R(x) \in R_y$ we must have $\mathfrak{h} \in L$. This follows from the fact that $dist(y, f_R(x)) = \langle \mathfrak{h}, \epsilon\rangle$ and that $dist(x,y) \in \langle L, V^*\rangle$. So indeed, L must be closed under postfixes if Monotonicity holds. Reversing the argument shows that if L is closed under postfixes, the relation defined is Monotone. ⊣

Remains to see what Tightness amounts to. Here assume that $y < f_R(x)$. Let $y \in x\langle \mathfrak{u}|\mathfrak{d}\rangle$, where $\mathfrak{u} \in L$ and $\mathfrak{d} \neq \epsilon$. Assume now that $z > y$. If $z \geq x$ we must have $z \leq f_R(x)$. (In the other case this automatically satisfied.) Now, $z = \mathfrak{d}^T \cdot \mathfrak{e}$ for some $\mathfrak{d}^T \cdot \mathfrak{e} \in L$, and so, by suffix closure, $\mathfrak{e} \in L$. Then $z \in x\langle \mathfrak{u}|\mathfrak{d}\rangle\langle \mathfrak{d}^T \cdot \mathfrak{e}|$ and thus $z \in x\langle \mathfrak{u} \cdot \mathfrak{e}|$. $z \leq f_R(x)$ is then equivalent with $\mathfrak{u} \cdot \mathfrak{e} \in L$. Let now $L * L = \{\mathfrak{d} \cdot \mathfrak{e} : (\exists a \in V)(\mathfrak{d} \cdot a, a \cdot \mathfrak{e} \in L)\}$.

Theorem 4.7.8 *An abstract relation is a tight command relation iff it is a left oval $\langle L|V^*\rangle$ such that $L \supseteq V$, L is closed under subwords and $L * L \subseteq L$.* ⊣

Examples. Tight relations are defined by the following languages. (i) $\{\epsilon\} \cup V \cup W^*$ for some $W \subseteq V$. This language defines a relation $\mathfrak{R}$ via $f_R(x) =$ *the smallest in a continuous row of W properly above x.* (ii) $\{\epsilon\} \cup V \cup (V - W)^* \cup (V - W)^* \cdot W$. This is the standard type: $f_R(x) =$ *the first W strictly above x.* (iii) $L =$ all subwords of $a_1 \cdot \ldots \cdot a_n$, where $V = \{a_1, \ldots, a_n\}$. The characteristic function of this relation is rather tricky to determine. However, this last example shows that tight relations can be rather odd in their behaviour.

It is not hard to check that the following laws hold

$$\langle L_1|V^*\rangle \cup \langle L_2|V^*\rangle = \langle L_1 \cup L_2|V^*\rangle$$

$$\langle L_1|V^*\rangle \cap \langle L_2|V^*\rangle = \langle L_1 \cap L_2|V^*\rangle$$

Moreover, if L_1 and L_2 are prefix (postfix) closed, then so are their intersection and join. As for composition the law

$$\langle L_1|V^*\rangle \circ \langle L_2|V^*\rangle = \langle L_1 \cdot L_2|V^*\rangle$$

does not necessarily hold. It does hold, however, when both are monotone, that is, suffix closed.

Theorem 4.7.9 *Regular command relations are closed under intersection and union; the monotone, regular command relations are in addition closed under composition.* ⊣

Chapter 5

Boolean Grammars

5.1 Context Free Grammars

§ 5.1 is based on (Harrison, 1978), (Becker and Walter, 1977) and (**?**). A context-free **rewrite system** is a pair $\mathbb{R} = \langle V, R\rangle$ where V is a non-empty set, the so-called **vocabulary** and $R \subset V \times V^*$ a finite set, the set of **rules**. A rule is commonly written as $\rho = A \to \Gamma$ where $A \in V$ and $\Gamma \in V^*$. ρ is said to be **n-ary** if Γ is a string of length n. In that case we also say that the **productivity** of ρ is $n - 1$; the productivity is denoted by p_ρ. Note that p_ρ can be any number ≥ -1. If $p_\rho \leq 0$, ρ is called **unproductive**. A context free rewrite system stands in correspondence with the set of V-**strings** as well as the set of (ordered) V-trees it generates. We define $\mathbb{R} \gg \mathfrak{T} = \langle \mathbb{T}, \sqsubset, \ell\rangle$ as follows. $\mathbb{R} \gg \mathfrak{T}$ if for every $x \in T$ which is not a leaf and $x^\downarrow = y_1 \dots y_n$ the string of daughters of x with $y_i \sqsubset y_{i+1}$ we have $\ell(x) \to \ell(y_1)\dots\ell(y_n) \in R$. If $\mathbb{R} \gg \mathfrak{T}$ then $\mathbb{R}$ is said to **generate** $\mathfrak{T}$. The trees generated by $\mathbb{R}$ is called the **yield** of $\mathbb{R}$; we symbolize it by $\mathrm{Yd}(\mathbb{R})$. Notice that all one-element trees are generated. The strings generated by $\mathbb{R}$ are defined thus. All $v \in V^1$ are generated. Furthermore, if $\Gamma_l v \Gamma_r$ is a string generated by $\mathbb{R}$ and $v \to \Delta \in R$, then $\Gamma_l \Delta \Gamma_r$ is a string generated by $\mathbb{R}$. Note that if we use $\rho = v \to \Delta$ to rewrite v in this way then $\sharp(\Gamma_l \Delta \Gamma_r) = \sharp(\Gamma_l v \Gamma_r) + p_\rho$. Note that the lengths of strings do not have to increase; we can only say that $\sharp(\Gamma_l \Delta \Gamma_r) \geq \sharp(\Gamma_l v \Gamma_r) - 1$. We write $L(\mathbb{R})$ for the set of strings generated by $\mathbb{R}$ and call it the **language of** $\mathbb{R}$. Notice that $L(\mathbb{R})$ is the set of all string cuts of trees generated by $\mathbb{R}$. Notice also that $\mathbb{R}$ also generates the subtree consisting of all elements x with $x \geq c$ for some $c \in C$, i. e. the set $\uparrow C$. Furthermore, C turns out to be the set of leaves of this tree. So, $L(\mathbb{R})$ is the set of minimal string cuts of trees generated by $\mathbb{R}$. We also write $\mathbb{R} \gg \vec{v}$ for $\vec{v} \in L(\mathbb{R})$.

Context-free grammars are rewrite systems in which the symbols from which to start a derivation and the symbols with which to end derivations are fixed. Namely, select from V a subset S and a subset T. Then $\mathbb{G} = \langle V, S, T, R\rangle$ is a **context-free grammar** if $\langle V, R\rangle$

is a rewrite system and no rule exists of the form $t \to \Gamma$ where $t \in T$. We follow the usual convention to denote terminal symbols by lower case letters and non-terminal symbols by upper case letters. If a symbol is undetermined in this respect we use lower case Greek letters. We say that $\mathbb{G}$ generates the ordered labelled tree $\mathfrak{T}$ if generates the tree as a rewrite system and, moreover, $\ell(r) \in S$ and $\ell(l) \in T$ if l is a leaf. We also write $\mathbb{G} \gg \mathbb{T}$. The **language generated** by $\mathbb{G}$ means either the subset of V^* generated by $\mathbb{G}$ – in which case it is the set of cuts of trees generated by the grammar – or it can mean the set of all strings in T^* generated by $\mathbb{G}$ – in which case it means the set of minimal cuts of trees generated by the grammar. We distinguish the two (whenever necessary) by $L^n(\mathbb{G})$ and $L^t(\mathbb{G})$. A language L is called **context free** if $L = L^t(\mathbb{G})$ for some context free $\mathbb{G}$.

Context-free grammars are quite a general concept. For every language generable by a cfg there are infinitely many cfgs generating this language. Moreover, there are bad and good grammars to write for generating a language. We will be concerned to see how good grammars can be in the general case. Before we do so we have to show that our definition of a context-free grammar is the same as the usual one. Namely, in standard definitions it is required that S contains only one symbol. Now, given a grammar in our sense, we add a new symbol $\aleph$ and new rules $\aleph \to \Gamma$ for each rule $W \to \Gamma \in R$ with $W \in S$. We then declare $\aleph$ the only start symbol; that is, we form the grammar

$$\langle V \cup \{\aleph\}, \{\aleph\}, T, R \cup \{\aleph \to \Gamma | (\exists W \in S)(W \to \Gamma \in R)\}\rangle$$

This grammar has a single start symbol and – with the exception of the root label – it generates exactly the same ordered labelled trees. Notice that in this new grammar the start symbol does not occur to the right of a rule. So there is now a symmetry between start symbols and terminal symbols because the latter may not appear to the left of a rule. Notice that the proofs also reveals that the languages generable by cfgs are closed under union. Namely, take two grammars $\mathbb{G} = \langle V, S, T, R\rangle$ and $\mathbb{H} = \langle V', S', T, R'\rangle$ over the same alphabet. One can arrange it that the non-terminals of both grammars are disjoint sets. Then let $\mathbb{G} \cup \mathbb{H} = \langle V \cup V', S \cup S', T, R \cup R'\rangle$. It is routine to check that a tree is generated by $\mathbb{G} \cup \mathbb{H}$ iff it is generated by one of $\mathbb{G}$ or $\mathbb{H}$.

There are two quite obvious deficiencies that a context free grammar can have. These are exemplified by the grammars over the vocabulary $\{S, A, a\}$ with set of terminals $\{a\}$. The first has the rules $S \to S, S \to a, A \to a$. Here the symbol A and the rule $A \to a$ although technically present can never be used in an actual derivation from the grammar. The second grammar only has the rule $S \to Sa$. Here there is no possibility to end a derivation. A grammar is called **normal** if it has neither of these defects.

Theorem 5.1.1 *For each grammar $\mathbb{G}$ there exists a normal grammar $\mathbb{G}^n$ generating exactly the same set of trees.*

Proof. Define the notion of *reachable* and *groundable* symbols. $\sigma \in V$ is **reachable** if either $\sigma \in S$ or there is a rule $W \to \Gamma$ such that W is reachable and $\sigma \in \Gamma$. σ is **groundable**

if either $\sigma \in T$ or there is a rule $\sigma \to \Gamma$ such that all $\alpha \in \Gamma$ are groundable. Let $S^n = S$, V^n be the set of all $\sigma \in V$ which are both reachable and groundable, $T^n = T \cap V^n$ and, finally, $R^n = R \cap V^n \times V^{n*}$. And $\mathbb{G}^n = \langle V^n, S^n, T^n, R^n \rangle$. It is not hard to see that if $\mathbb{G}^n \gg \mathfrak{T}$ then $\mathbb{G} \gg \mathfrak{T}$ as well, because the latter has more rules and more symbols. Now let $\mathbb{G} \gg \mathfrak{T}$. Then by induction on the depth it is shown that all labels $\ell(x)$ for $x \in T$ are reachable. Likewise, by induction from bottom to top it is shown that all $\ell(x)$ are groundable. Then if $x \in T$ is not a leaf and $y_1 \ldots y_n$ is the sequence of its daughters then $\ell(x) \to \ell(y_1) \ldots \ell(y_n) \in R$ and all symbols are in V^n; hence $\ell(x) \to \ell(y_1) \ldots \ell(y_n) \in R^n$. ⊣

In normal grammars a symbol is a terminal symbol iff it does not occur to the left in a production. We have seen that we can arrange it that a symbol is a start symbol iff it does not occur to the right of a rule. Hence grammars can be identified with context-free rewrite systems in which the terminal and the start symbols are left implicit, that is, in which these sets are recoverable from the rules. From now on we assume all grammars to be normal. There are other, more serious 'deficiencies' a grammar might have. Particularly problematic are the unproductive rules, namely $A \to \epsilon$ and unary rules $A \to B$. It is not difficult to understand that one wants to get rid of such rules in a grammar. One actually can get rid of them but at the cost of writing a grammar that does not generate the same trees. However, as we already explained, it is constituent structures rather than trees that we need to keep invariant under our manipulations; this is so, because it is in principle unobservable from which tree structure the observed constituent structures of sentences arise. This statement inasmuch as it arises naturally here, is one of formal content. It is quite controversial in linguistics how far removed the analysis of sentence structure needs to be from the visible structure, that is, the constituent structure. In GB it is often claimed that there can be principles reasons to choose one rather the other. But the arguments as far as I have seen are only of intrinsical nature, stemming from considerations about a (or *the*?) theory of grammar, rather than from the pure and simple concern to explain the data at hand. They might be right in saying that the formal reduction has no psychological basis but they are wrong in suggesting that this can be seen from judging the data alone. Psychological claims cannot be proved other than by psychological experiment.

The main result of this paragraph is a proof that for every grammar there is an effective algorithm to derive a grammar without unproductive transitions which nevertheless has the same constituent structures and lets itself be represented faithfully by an associated rewrite system. The latter property is of technical importance, while the former are of wider importance for the application of formal language theory. First we consider the elmination of unproductive rules. We introduce the *skeletal type* of a grammar. For a symbol $\sigma \in V$ we let $\mathrm{sk}(\sigma) = n$ if σ is non-terminal and $\mathrm{sk}(\sigma) = t$ if σ is terminal. For a rule $\rho = X \to \sigma_1 \ldots \sigma_n$ we let $\mathrm{sk}(\rho) = \langle \mathrm{sk}(\sigma_1), \ldots, \mathrm{sk}(\sigma_n) \rangle$. So, the **skeleton** $\mathrm{sk}(\rho)$ of a rule ρ tells us which elements of the production of the rule are terminal and which of them are not. The **skeletal type** of $\mathbb{G}$ is defined by $\mathrm{sk}(\mathbb{G}) = \{\mathrm{sk}(\rho) | \rho \in R\}$. For example, a grammar is in **Chomsky Normal Form** if it is of skeletal type $\{\langle t \rangle, \langle n \rangle, \langle n, n \rangle\}$, that is, a rule is either of the form $A \to a$, a terminal or of the form $A \to B$ with B non-terminal or else of the form $A \to BC$, B, C non-terminal. (This ignores the possibility to have the rule $S \to \epsilon$ which allows to generate empty strings, but for proper languages this is excluded; simply assume that none of the grammars considered here allow to generate the empty string.) We will show that there is an algorithm transforming a grammar $\mathbb{G}$ into a grammar $\mathbb{G}^c$ which allows the same constituent structures but is of skeletal type contained in the type $\{\langle t\rangle, \langle n, n\rangle, \langle n, n, n\rangle, \ldots\}$. To this end we need to eliminate certain rules. The first simplification consists in introducing a nonterminal A for each terminal $a \in V$ together with a rule $A \to a$. These extra nonterminal symbols are called **preterminal** since it is the case that they can only cover leaves in the generated trees. Furthermore, if $W \to \Gamma$ is a rule in $\mathbb{G}$ containing some non-terminals we replace any non-terminal a in Γ by its corresponding preterminal. For example, the rule $X \to aZbXc$ will be replaced by the rule $X \to AZBXC$ with the rules $A \to a, B \to b, C \to c$ added etc. It is straightforward to check that this operation leaves the set of constituent structures unharmed. Observe namely that in the constituent structure any leaf i is covered by the preleaf $\{i\}$, so that any single symbol in a generated string is in fact treated as a constituent. The new grammar indeed generates it as a constituent and so comes closer to the constituent structures. Next we attack the empty productions. Suppose our grammar contains an empty production $W \to \epsilon$. Let $\Delta_1, \ldots, \Delta_n$ be the (possibly empty) enumeration of Δ_i such that $W \to \Delta_i \in R$. Now for any rule $\rho = Z \to X_1 \ldots X_r$ we let ρ^W be the set of rules obtained by replacing the occurrences of W as $W = X_i$ by any of the Δ_j. Let $R^W = \bigcup\langle \rho^W | \rho \in (R-(W \to \epsilon))\rangle$. Notice that if there are no Δ_i the set ρ^W is empty. R^W does not contain the empty production $W \to \epsilon$; it does contain still the other empty productions, but no more. It generates the same strings and the same constituent structures. We do this procedure for each empty production – thus eliminating them all. In the next step we get rid of the productions of the type $W \to X$. First we throw away all productions $W \to W$. Then we do a construction which is quite similar to the one above. In every rule $\rho = Z \to Y_1 \ldots Y_r$ we replace all occurrences of $W = Y_j$ by a Δ such that $W \to \Delta$ is a rule and Δ has at least two symbols. After adding all such rules, $W \to X$ can be retracted. Notice that the remaining grammar has one unary production less and produces the same constituent structures.

The next property we want to monitor is the correspondence with a rewrite system. Notice that we have already spoken of the possiblity to let terminal and initial symbols be implicitly defined by the rules. The initial symbols cannot appear on the right, the terminal symbols cannot appear on the left. Still, in a nonstandard grammar the mixed apprearance of terminal and initial symbols in the right hand side of a rule causes concern if we aim at eliminating the terminal symbols altogether. At present this may appear unmotivated, but let us grant that one may be interested in this question. The question can be put as follows. Suppose that we have a language over T and a context-free grammar $\mathbb{G}$. Suppose that we drop all terminal symbols in the rules, suppose we forget what is the start symbol – under what circumstances is $\mathbb{G}$ fully recoverable? For example, let $\mathbb{G}$ be the following grammars

$\mathbb{G}$		$\mathbb{H}$	
$S \to AaA$	$A \to SaS$	$S \to AaaA$	$A \to SaS$
$A \to a$		$A \to aa$	

If all terminals are dropped we get in both cases the grammar $S \to AA, A \to SS$. However, none of the grammars is standard. Namely, the assumpmtion of standardness lets us locate the terminal symbols in a possible rule; for it can only be unary. However, dropping terminals from a rule lets no trace of that rule, so we cannot recover it. The problem is that we do not know which symbols admit an expansion into a terminal symbol. Technically, this can be overcome by splitting a grammar into *syntax* and *lexicon*. We define here a **lexicon** as a pair $\Lambda = \langle N, T, \lambda \rangle$ where T is a set of terminals, N a set of nonterminals and $\lambda : T \to N$ a map from T to N. Precisely this split is made in current syntax; there is the *grammar* and there is the *lexicon*. The grammar if properly written lets us eliminate all reference to terminals and so reduces to a pure cf rewrite system. This rewrite system together with the lexicon lets us reconstruct the 'ordinary' cfg.

Without the lexicon, however, it is necessary to have a distinction between those nonterminals that can occur as preterminals and those which cannot. This, however, is an important question. Let us say that a labelled tree $\mathfrak{F}$ is the **frame** of $\mathfrak{T}$, if it results from $\mathfrak{T}$ by dropping all leaves. The question arises how we can determine what *frames* the grammar generates, given only its rewrite system. To be able to answer this we need to know which of the trees generated by the rewrite systems are completed, that is, have leaves that can be expanded into terminals in the original grammar. In other words, we need to be able to distinguish preterminal systems. This, however, is not possible without assuming that our grammar is of a special form. Namely, it must be such that a nonterminal the choice of appearing either exclusively as a preterminal or exclusively as a non-preterminal. This whole exercise is principally the same as with the terminal symbols, now done with the preterminals. If a grammar is organized in this way, the preterminals function as classifiers of the terminals. They denote classes of syntactically indistinguishable terminals; whereby we mean to say that if $P \to a$ and $P \to b$ are two expansions of the preterminal P, then a and b can be substituted for each other in all contexts. Clearly, if a grammar has a non-terminal N that may also occur as a preterminal, we can get around the problem by splitting N into the twins N_p, N_n. All rules $N \to t$ are replaced by $N_p \to t$ and all rules $N \to \Delta$ are replaced by the set of rules of the form $N_n \to \widehat{\Delta}$ where $\widehat{\Delta}$ arises from replacing N by either of the twins N_p, N_n.

Definition 5.1.2 *We call $\mathbb{G}$ in* ***standard form*** *if*

$$\mathrm{sk}(\mathbb{G}) \subseteq \{\langle t \rangle, \langle n, n \rangle, \langle n, n, n \rangle, \ldots\}$$

and every nonterminal N is either strictly preterminal or strictly non-preterminal.

Standard grammars have the advantage of generating constituent structures directly as trees. If $\langle \mathbb{T}, \sqsubset, \ell \rangle$ is a labelled ordered tree generated by a standard grammar then the constituent structure of this tree is basically the tree itself (just drop the superfluous labels). Furthermore, a cf rewrite system $\langle N, \mathbb{R} \rangle$ based on nonterminals lets us recover $\mathbb{G}$ if $\mathbb{G}$ is standard.

Theorem 5.1.3 *Any grammar $\mathbb{G}$ with $S \to \epsilon \notin R$ can be reduced algorithmically to standard form $\mathbb{G}^s$ such that both $\mathbb{G}$ and $\mathbb{G}^s$ admit the same constituent structures.* ⊣

5.2 Perfect Grammars

Another important concept is that of an *invertible grammar*.

Definition 5.2.1 *A grammar $\mathbb{G}$ is invertible if for any pair of rules $A \to \Gamma$ and $B \to \Gamma$ in R we have $A = B$.*

Invertibility is quite an important property. Namely, suppose we have a standard, invertible $\mathbb{G}$ and a constituent structure $\sigma = \langle \vec{v}, \mathfrak{P} \rangle$ generated by $\mathbb{G}$. Then there is a unique way to extend this constituent structure to a tree generated by $\mathbb{G}$. Namely, by the fact that $\mathbb{G}$ is standard, the constituent structure belongs to a tree whose minimal cut is $\vec{v}$. Thus the problem is to introduce the labels at the interior nodes. But here we can proceed inductively; suppose that $x \in \mathfrak{P}$ and that x immediately dominates $y_1, \ldots, y_n$ in $\langle \mathfrak{P}, \subseteq \rangle$ with $y_i \sqsubset y_{i+1}$. Suppose inductively that the labels $\ell(y_1), \ldots, \ell(y_n)$ have been assigned. Then there is at most one $\ell(x)$ to make $\ell(x) \to \ell(y_1) \ldots \ell(y_n)$ a rule of $\mathbb{G}$. That there is at least one way follows from the assumption that $\vec{v} \in L(\mathbb{G})$, so there is a tree with minimal cut $\vec{v}$.

Proposition 5.2.2 *Let $\mathbb{G}$ be standard and invertible and $\langle \vec{v}, \mathfrak{P} \rangle$ a constituent structure of $\mathbb{G}$. Then there is a unique tree generated by $\mathbb{G}$ with constituent structure $\langle \vec{v}, \mathfrak{P} \rangle$.*

Proof. By standardness, the tree based on the constituent structure is the only tree that assigns this constituent structure to $\vec{v}$. The labelling is unique by the fact that $\mathbb{G}$ is invertible. ⊣

Theorem 5.2.3 *There is an algorithm transforming a grammar $\mathbb{G}$ into an invertible grammar $\mathbb{G}^i$ such that both admit the same constituent structures.*

Proof. For simplicity we assume that $\mathbb{G}$ has no unproductive rules. We take $T^i = \{\{t\} | t \in T\}$ but $V^i = T^i \cup N^i$ with $N^i = 2^N - \{\emptyset\}$ with $N = V - T$. So, while the terminals of $\mathbb{G}^i$

are in effect the terminals of $\mathbb{G}$, the nonterminals of $\mathbb{G}^i$ are the sets of nonterminals of $\mathbb{G}$. Intuitively, $\{A_1, \ldots, A_n\}$ is a label that if assigned to x means *has label* A_1 *or* A_2*...or* A_n. Then $S^i = \{A | A \cap S \neq \emptyset\}$. Furthermore, as rules we take all rules of the form

$$\mathfrak{A} \to \mathfrak{B}_1 \ldots \mathbb{G}_n$$

such that whenever $A \in \mathfrak{A}$ there exist $B_i \in \mathfrak{B}_i$ such that $A \to B_1 \ldots B_n \in R$. (Notice that the B_i may also be terminals in which case $\mathfrak{B}_i = \{b_i\}$, by definition of T^i.) By construction, $\mathbb{G}^i = \langle V^i, S^i, T^i, R^i \rangle$ is invertible. For if the $\mathfrak{B}_1, \ldots, \mathfrak{B}_n$ are fixed, the set $\mathfrak{A}$ is uniquely determined. It remains to be seen, however, that $\mathbb{G}^i$ has the same constituent structures as $\mathbb{G}$. Consider to this effect a labelled tree $\mathfrak{T} \ll \mathbb{G}$. By induction from bottom to top we show that there is a $\mathfrak{U} = \langle \mathbb{T}, \sqsubset, \mu \rangle$ such that $\mathbb{G}^i \gg \mathfrak{U}$. On the leaves, let $\mu(l) = \{\ell(l)\}$. Suppose then that x is not a leaf and that $\mu(y_i)$ has been defined for all $y_i \in x^{\downarrow}$. Then, inductively, it holds that $\ell(y_i) \in \mu(y_i)$ and moreover $\ell(x) \to \ell(y_1) \ldots \ell(y_n) \in R$. Then the set $\mathfrak{A}$ of all A such that $A \to B_1 \ldots B_n \in R$ for some $B_i \in \mu(y_i)$ is not empty and so $\mu(x) = \mathfrak{A}$ is a good candidate. Eventually, μ is defined over the whole tree and this gives the desired $\mathbb{U}$. Conversely, assume $\mathbb{U} \ll \mathbb{G}^i$ is given. We now want to produce a ℓ such that $\mathbb{G} \gg \langle \mathbb{T}, \sqsubset, \ell \rangle$. Here we proceed top to bottom. We choose a $A \in \mu(r)$ and let $\ell(r) = A$. Inductively, assume $\ell(x) = A$ that $x^{\downarrow} = y_1 \ldots y_n$ and that $\mu(y_i) = \mathfrak{B}_{\mathrm{i}}$. By construction of $\mathbb{G}^i$ there must be $B_i \in \mathfrak{B}_i$ such that $A \to B_1 \ldots B_n \in R$. Define $\ell(y_i) = B_i$. Proceed inductively. Eventually, ℓ is defined over the whole tree. Since for the leaves $\ell(l) \in \mu(l)$ and $\mu(l) = \{t_l\}$, we have by the identification $t \mapsto \{t\}$ that both trees have the same minimal string cut. This proves the theorem. ⊣

After having reduced the grammars to standard, invertible form there is a final reduction that one can make. Consider the following grammar.

$$\begin{array}{ll} S \to AA & A \to CD \\ S \to AB & B \to DC \\ S \to BA & C \to c \\ S \to BB & D \to d \end{array}$$

In this grammar the two symbols A, B although extensionally different in the sense that they develop into different trees are intensionally identical in that they can be substituted for each other in any right hand side of a rule. It is not hard to see that the difference between A and B is syntactically irrelevant. We can namely write a simpler grammar in which A and B are identified, and the same trees are generated.

$$\begin{array}{ll} S \to AA & C \to c \\ A \to CD & D \to d \\ A \to DC & \end{array}$$

Both grammars are invertible and standard. Yet the second is obviously simpler. This motivates the following definition.

Definition 5.2.4 *Let $A, B \in V - T$. A and B are called* ***indistinguishable*** *in $\mathbb{G}$ if $A \in S \Leftrightarrow B \in S$ and for every rule $\rho = X \to \Gamma \in R$ also $X \to \Gamma' \in R$ where Γ' results from Γ by replacing some occurrences of A by B and some occurrences of B by A. A grammar is* ***refined*** *if for every pair A, B if A is indistinguishable from B then $A = B$.*

Lemma 5.2.5 *Let $\mathbb{G}$ be normal, standard and invertible and A, B be nonterminals. Then A is indistinguishable from B iff for every pair $\vec{w}_1, \vec{w}_2$ of constituents such that $\vec{w}_1$ is of category A and $\vec{w}_2$ of category B and any constituent $\vec{v}$ $\mathbb{G} \gg \vec{v} \Leftrightarrow \mathbb{G} \gg \vec{v}'$ where $\vec{v}'$ results from $\vec{v}$ by replacing a constituent occurrence of $\vec{w}_1$ by $\vec{w}_2$ or a constituent occurrence of $\vec{w}_2$ by $\vec{w}_1$*

Proof. Straightforward from the fact that $\vec{w}_1$ occurs as a constituent iff it occurs as a constituent with label A and $\vec{w}_2$ occurs as a constituent iff it occurs as a constituent with label B. ⊣

Now let $\approx$ be an equivalence relation such that $A \approx B$ implies that A and B are indistinguishable. Let π be the canonical map $\pi : V \to V/{\approx}$, where $V/{\approx}$ is the set of $\approx$-equivalence classes. We put $V^\pi = V/{\approx}$, $S^\pi = \{X/{\approx} | X \in S\}$, $T^\pi = T$, $R^\pi = \{x/{\approx} \to \sigma_1/{\approx} \dots \sigma_n/{\approx} | X \to \sigma_1 \dots \sigma_n \in R\}$. Then put $\mathbb{G}^\pi = \langle V^\pi, S^\pi, T^\pi, R^\pi \rangle$. We call π a **refinement** map and $\mathbb{G}^\pi$ a **refinement** of $\mathbb{G}$. If $\mathbb{G}$ admits no nontrivial refinements $\mathbb{G}$ is called **refined**. If $\approx$ is defined via $A \approx B$ iff A and B are indistinguishable then $\mathbb{G}^\pi$ is not necessarily refined; however, since this grammar has less symbols, a suitable iteration of this process will yield a refined grammar.

Theorem 5.2.6 *There is an algorithm transforming any grammar $\mathbb{G}$ into a refined grammar $\mathbb{G}^r$ which generates the same trees via the canonical translation $\sigma \mapsto \sigma/{\approx}$ as $\mathbb{G}$. Moreover, if $\mathbb{G}$ is invertible, so is $\mathbb{G}^p$.*

Proof. For a tree $\mathfrak{T} = \langle \mathbb{T}, \sqsubset, \ell \rangle$ we put $\mathfrak{T}^r = \langle \mathbb{T}, \sqsubset, \ell^r \rangle$ with $\ell^r(x) = \ell(x)/{\approx}$. It is straightforward to check that if $\mathbb{G} \gg \mathfrak{T}$ then $\mathbb{G}^r \gg \mathfrak{T}^r$. Moreover, if $\mathbb{G}^r \gg \mathfrak{T}^r$ then there exists a tree $\mathbb{U}\langle \mathbb{T}, \sqsubset, \mu \rangle$ with $\mathbb{U}^r = \mathfrak{T}^r$ and $\mathbb{G} \gg \mathbb{U}$. It remains to determine μ. Obviously, for leaves $\mu(l) = \ell(l)$ satisfies the requirements. Furthermore, let x be not a leaf and $x^\downarrow = y_1 \dots y_n$. Assume that $\mu(y_i)$ is defined and $\mu(y_i)/{\approx} = \ell(y_i)/{\approx}$. By assumption, $\ell(x)/{\approx} \to \ell(y_1)/{\approx} \dots \ell(y_n)/{\approx} \in R^r$ and so $X \to \mu(y_1) \dots \mu(y_n) \in R$ for some $X \approx \ell(x)$. Put $\mu(x) = X$. Finally, μ is defined over the whole tree and $\mu(r) \approx \ell(r)$. Since $\ell(r) \in S$, $\mu(r) \in S$ and this shows $\mathbb{G} \gg \mathbb{U}$. This proves the first assertion. For the second let $A/{\approx} \to \Gamma/{\approx}, B/{\approx} \to \Gamma/{\approx} \in R^p$. Then for some $\Delta_1, \Delta_2 \approx \Gamma$, some $\widehat{A} \approx A$ and $\widehat{B} \approx B$ we have $\widehat{A} \to \Delta_1$ and $\widehat{B} \to \Delta_2$. Now, by definition of indistinguishability, $\widehat{A} \to \Gamma, \widehat{B} \to \Gamma \in R$ and so by the fact that $\mathbb{G}$ is invertible, $\widehat{A} = \widehat{B}$, which means $A/{\approx} = B/{\approx}$. ⊣

Definition 5.2.7 *A grammar $\mathbb{G}$ is* **perfect** *if it is normal, standard, invertible and refined.*

Theorem 5.2.8 *For every $\mathbb{G}$ there exists a perfect $\mathbb{G}^p$ such that* $\mathrm{Cst}(\mathbb{G}^p) = \mathrm{Cst}(\mathbb{G})$.

Proof. Reduce $\mathbb{G}$ first to normal and then to standard form. Make $\mathbb{G}$ invertible and refine it. The last step leads to a perfect grammar, since invertibility is not lost. ⊣

Perfect grammars have interesting properties. The most remarkable is the following.

Proposition 5.2.9 *There exists up to renaming of the nonterminals exactly one perfect grammar $\mathbb{P}$ generating* $\mathrm{Cst}(\mathbb{P})$.

Proof. Let $\mathbb{G}$ be a perfect grammar such that $\mathrm{Cst}(\mathbb{G}) = \mathrm{Cst}(\mathbb{P})$. Define a relation $\tau \subseteq N_{\mathbb{P}} \times N_{\mathbb{G}}$ from the nonterminals of $\mathbb{P}$ to the nonterminals of $\mathbb{G}$ as follows. Given a $\langle \vec{v}, \mathfrak{P} \rangle \in \mathrm{Cst}(\mathbb{P})$ and a constituent $\vec{w}$ of $\vec{v}$ in this structure. Then by Proposition 5.2.2 $\vec{w}$ has a unique label W in $\mathbb{P}$ and a unique label X in $\mathbb{G}$. In this case we let $W \tau X$. So, τ collects all instances of W, X where W is the label of a constituent in $\mathbb{P}$ and X the label of a constituent in $\mathbb{G}$. Suppose now that $W \tau X_1, X_2$. Then there are constituents $\vec{w}_1, \vec{w}_2$ in some structures such that both $\vec{w}_1, \vec{w}_2$ have label X in $\mathbb{P}$ but $\vec{w}_1$ has label X_1 in $\mathbb{G}$ and $\vec{w}_2$ has label X_2. Then because the strings $\vec{w}_1, \vec{w}_2$ have the same label in $\mathbb{P}$ they can be substituted for each other in each string $\vec{v}$ if they occur as constituents. On the other hand, since this is so and $\mathbb{G}$ is normal, standard and invertible, we can invoke Lemma 5.2.5 to prove that $X_1 \approx X_2$ and hence $X_1 = X_2$, since $\mathbb{G}$ is reduced. This proves that there is a function $\tau : N_{\mathbb{P}} \to N_{\mathbb{G}}$ such that $\mathbb{P} \gg \mathfrak{T} \Leftrightarrow \mathbb{G} \gg \mathfrak{T}^{\tau}$ and $\tau \restriction T = id$. By reversing the roles we can show that there is a similar function $\sigma : N_{\mathbb{P}} \to N_{\mathbb{G}}$ with $\mathbb{P} \gg \mathfrak{T} \Leftrightarrow \mathbb{G} \gg \mathfrak{T}^{\sigma}$. Now $\mathbb{P} \gg \mathfrak{T} \Leftrightarrow \mathbb{P} \gg \mathfrak{T}^{\tau\sigma}$ and $\mathbb{G} \gg \mathfrak{T} \Leftrightarrow \mathbb{G} \gg \mathfrak{T}^{\sigma\tau}$, and $\sigma \circ \tau \restriction T = id, \tau \circ \sigma \restriction T = id$. From the invertibility of $\mathbb{P}$ it follows $\tau \circ \sigma = id$ and from the invertibility of $\mathbb{G}$ it follows that $\sigma \circ \tau = id$, which proves the theorem. ⊣

Theorem 5.2.10 *It is decidable for two grammars $\mathbb{G}$ and $\mathbb{H}$ over a given set of terminal symbols whether or not* $\mathrm{Cst}(\mathbb{G}) = \mathrm{Cst}(\mathbb{H})$.

Proof. First, it can be assumed that both $\mathbb{G}$ and $\mathbb{H}$ are perfect; otherwise reduce $\mathbb{G}, \mathbb{H}$ to perfect form. Then $\mathrm{Cst}(\mathbb{G}) = \mathrm{Cst}(\mathbb{H})$ is decidable by Proposition 5.2.9 because if they generate the same constituent structures they must in fact be isomorphic. But whether two finite structures are isomorphic, is decidable. ⊣

This result is also proved essentially in (**?**), though the results do not explicitly appear in that form due to the use of Greibach Normal Form. In the quoted paper the notion of a *bisimulation* is used, which is technically somewhat simpler than the notion of syntactical indistinguishability but intuitively more demanding. used instead.

5.3 Regular Languages

Definition 5.3.1 *A language $L \subseteq V^*$ is called* **regular** *if it can be produced from the sets $\{\epsilon\}$, $\{a\}$, $a \in V$, with the help of the operations $\cup, \cdot$ and the Kleene-star $-^*$.*

The following are regular languages. V^*, the set of all strings, $a^* = \{a^n | n \in \omega\}$, $(V - \{a\})^* \cdot a$. Note that we drop the brackets from $\{a\}$ and write a instead. Also, we write a^n for the n-fold concatenation of a with itself.

There are two characterization theorems for regular languages, one in terms of the grammars generating them and the other in terms of machines that accept these languages. We will not prove these results; proofs can be found for example in (Harrison, 1978). Call a context-free grammar **right linear** if it has type $\{\langle t\rangle, \langle n\rangle, \langle t, n\rangle\}$, that is, all the rules are of the form

$$A \to t, A \to B, A \to uB, \text{ for } t, u \in T, A, B \in N$$

Theorem 5.3.2 *A language $L \subseteq V^*$ is regular iff it is generated by a right linear grammar.* $\dashv$

A **finite state automaton** is a quintuple $A = \langle S, V, \tau, I, F\rangle$ where S is set, the set of *states*, I, F subsets of S, called the set of *initial* and the set of *final* states, V a set, the *vocabulary* and $\tau : V \times S \to S$ the *(direct) transition function*. From the direct transition function we can define the transition function $\widehat{\tau}$ inductively for $H \subseteq S$ via

$$\widehat{\tau}(\epsilon)(H) = H, \widehat{\tau}(\mathfrak{g} \cdot v)(H) = \tau(v)(\widehat{\tau}(\mathfrak{g})(H))$$

Notice that in contrast to the usual definition we have a *set* of initial states and that we define the transition function over sets of states rather than states. This is for convenience only. Now A **accepts** a string $\mathfrak{s}$ if $F \cap \widehat{\tau}(\mathfrak{s})(I) \neq \emptyset$, that is, if there exists an initial state $i \in I$ such that if $\mathfrak{s}$ is fed to the machine symbol by symbol the machine will at the end of the word reach a state $f \in F$. Call a language L **finite state recognizable** or simply **finite state** if there exists a finite state automaton A such that A recognizes $\mathfrak{s}$ iff $\mathfrak{s} \in L$.

Theorem 5.3.3 (Kleene) *A language is regular iff it is finite state.* $(\diamond)$

A usual finite state automaton has only a single i as a start symbol, that is, $\sharp I = 1$. There is a way to reduce an arbitrary automaton to one that has $\sharp I = 1$ and accepts the same language. Furthermore, an **indeterministic** automaton differs from a deterministic one in that $\tau : V \times S \to 2^S$ is a function not into S but the powerset S, so that in fact we have an indeterminism in the transition if $\sharp\tau(\sigma, s) \geq 2$. By introducing a different automaton, whose states are the sets of states of A we can produce a deterministic automaton $2^A = \langle 2^S, V, 2^\tau, I^d, F^d\rangle$ where $I^d = \{H | H \cap I \neq \emptyset\}$, $F^d = \{H | H \cap F \neq \emptyset\}$ and $2^\tau(\sigma)(H) = \bigcup\langle\tau(\sigma)(h) | h \in H\rangle$. Certainly, 2^A is deterministic; and it accepts exactly the strings which are accepted by A.

We will use these results to derive certain closure conditions on regular languages. First the following.

Theorem 5.3.4 *If L is regular, so is $V^* - L$. If L_1, L_2 are regular, so is $L_1 \cap L_2$.*

Proof. Suppose $A = \langle S, V, \tau, \{i\}, F\rangle$ is the automaton accepting L. Then $A^c = \langle S, V, \tau, \{i\}, S - F\rangle$ accepts $\vec{x}$ iff $\widehat{\tau}(\vec{x})(i) \in S - F$ iff $\widehat{\tau}(\vec{x})(i) \notin F$ iff A does not accept $\vec{x}$ iff $\vec{x} \notin L$. For the second claim observe that $L_1 \cap L_2 = V^* - ((V^* - L_1) \cup (V^* - L_2))$ and so, since regular languages are closed under union and under complementation, $L_1 \cap L_2$ is regular. ⊣

Theorem 5.3.5 *Suppose that L is regular. Then so are L^T, L^P and L^S.*

Proof. Let $A = \langle S, V, \tau, I, F\rangle$ accept A. Consider the automaton A^T which inverts the action of A: $A^T = \langle S, V, \tau^{-1}, F, I\rangle$ with $\tau^{-1}(\sigma)(H) = \{h | \tau(\sigma)(h) \in H\}$. A^T is not necessarily deterministic but that does not harm as we have seen. Now, by definition, A^T accepts $\vec{x}^T$ iff there is an initial state $f \in F$ such that $\vec{x}$ if fed in reverse order will yield a set T containing a final state $i \in I$. Hence, if $\vec{x}$ is fed in its proper order to A, starting with i, it will end in a set containing $f \in F$, hence accepting $\vec{x}$. This shows the first claim. Now consider the automaton $A^P = \langle S, V, \tau, I, F^P\rangle$ where F^P is the set of all h such that for some $\vec{t}$ $\widehat{\tau}(\vec{t})(h) \in F$. Then A^P accepts $\vec{x}$ iff there is a $i \in I$ and $h \in F^P$ such that $\widehat{\tau}(\vec{x})(i) = h$ iff there is $i \in I, f \in F$ and $\vec{y} \in V^*$ such that $\widehat{\tau}(\vec{x} \cdot \vec{y})(i) = f$ iff there is $\vec{y}$ such that A accepts $\vec{x} \cdot \vec{y}$ iff $\vec{x} \in L^P$. This shows the second claim. The third follows, because $L^S = ((L^T)^P)^T$, that is, suffixes are transposes of prefixes of the transpose of $\vec{x}$. It is also not difficult to produce an automaton accepting L^S. ⊣

Notice that in this proof it has payed off to have a set of initial states rather than a single such state. Recall the notation

$$L^{-1}M = \{m | (\exists\ell \in L)(\ell \cdot m \in M\}$$

$$LM^{-1} = \{\ell | (\exists m \in M)(\ell \cdot m \in L\}$$

Theorem 5.3.6 *If L is regular, so is LM^{-1}. If M is regular, so is $L^{-1}M$.*

Proof. Clearly, the second assertion follows from the first; namely, $L^{-1}M = (M^T(L^T)^{-1})^T$. So, if M is regularm so is M^T by Theorem refmirror, and so $M^T(L^T)^{-1}$ is regular, by the first statement. Hence $(M^T(L^T)^{-1})^T$ is regular, again by Theorem 5.3.5. Now for the first assertion. Let $A = \langle S, V, \tau, I, F\rangle$ accept L. Let $F^M = \{h|(\exists\mathfrak{m} \in M)(\widehat{\tau}(\mathfrak{m})(h) \in F\}$. Then $A^M = \langle S, V, \tau, I, F^M\rangle$ accepts $\mathfrak{s}$ iff $\widehat{\tau}(\mathfrak{s})(i) \in F^M$ for some $i \in I$ iff for some $\mathfrak{m} \in M$ and some $i \in I\,\widehat{\tau}(\mathfrak{s}\cdot\mathfrak{m})(i) \in F$ iff for some $\mathfrak{m} \in M$ A accepts $\mathfrak{s}\cdot\mathfrak{m}$ iff for some $\mathfrak{m} \in M$ $\mathfrak{s}\cdot\mathfrak{m} \in L$ iff $\mathfrak{s} \in LM^{-1}$. ⊣

5.4 Grammar Operations

Context free grammars can be manipulated in order to obtain new cfgs. Two operations are of particular significance. One is *cutting*. Given $\mathbb{G}$ and a set U of non-terminals, $\mathbb{G}-U$ is the result of dropping all non-terminals in U from $\mathbb{G}$. Thus, if $\mathbb{G} = \langle V, S, T, R\rangle$, then let

$$R - U = \langle A \to \Gamma | A \notin U, \Gamma \not\subseteq U\}$$

and then $\mathbb{G} - U = \langle V - U, S - U, T, R - U\rangle$. Cutting simply means to forbid certain nonterminals to occur. The second operation is *pairing*. With $\mathbb{G}_1 = \langle V_1, S_1, T, R_1\rangle$ and $\mathbb{G}_2 = \langle V_2, S_2, T, R_2\rangle$ we put

$$R_1 \times R_2 = \{\rho_1 \times \rho_2 | \rho_1 \in R_1, \rho_2\}$$

where $\rho_1 \times \rho_2$ is defined exactly when ρ_1 and ρ_2 have the same signature. Suppose that this is the case and let $\rho_1 = A_1 \to \Gamma_1, \rho_2 = A_2 \to \Gamma_2$; then $\rho_1 \times \rho_2 = \langle A_1, A_2\rangle \to \Gamma_1 \times \Gamma_2$ where the latter is defined according to $t \times t := t$ if t is terminal and $N_1 \times N_2 = \langle N_1, N_2\rangle$ if N_1, N_2 are nonterminal. Pairing of two grammars with identical alphabet of terminals results in something of an intersection of the languages. Namely, it can be shown that the trees generated by the pairing $\mathbb{G}_1 \times \mathbb{G}_2$ must match both generating rules of $\mathbb{G}_1$ and $\mathbb{G}_2$. This is intuitively clear, and a rigorous proof will be given later.

5.5 Going Boolean in Grammar

Syntax tries to abstract as much as possible from the actual words. It aims to produce a classificatory system of words that allows to abstract from the individual words to classes thereof. These classes represent words with identical syntactic behaviour; thus the classes group together words which are syntactically indistinguishable. Boolean grammars concern themselves exclusively with this latter classification and do not talk of words at all. This is the job of the lexicon. Notice that we have earlier been saying that an ordinary grammar can be decomposed into syntax and lexicon, where the latter is simply speaking a list telling us which word belongs to which group. Moreover, if we aim at completely disentangling syntax and lexicon the grammar needs to be prepared in *standard form* in order to be uniquely recoverable. Of course, since any grammar can be reduced to standard form keeping the constituent analysis constant, there is no real need to assume that the grammar is recoverable. For this shows that the outer form of the grammar is – as far as the constituent yield is concerned – somewhat arbitrary and it makes absolutely no sense to ask from which grammar the boolean grammar originated, except of course, it is syntactic trees we are after. Any pair of syntax and lexicon results in a standard grammar, and that is the grammar from which say that they originate.

After the lexicon being split from the grammar, we need to use only the classificatory system. This system we assume to be in boolean form. A priori we do not assume finiteness of the classificatory system, but for cfgs it necessarily will be (up to reduction). Such a classificatory scheme is best seen as a boolean algebra, or in fact a *mereology*. Notice namely, as we have pointed out earlier, that the 0 cannot hold of any element, so the classifiers that the syntax manipulates are all nonzero, and therefore the basic underlying classification scheme is a mereology rather than a boolean algebra.

Definition 5.5.1 *A* ***boolean rewrite system*** *is a pair* $\mathbb{R} = \langle \mathfrak{M}, R\rangle$ *where* $\mathfrak{M} = \mathfrak{B}_o$ *is a mereology,* R *a finite subset of* $M \times M^+$. $\mathbb{R}$ *is* ***context free*** *if* $\mathfrak{M}$ *is finite. A* ***booolean grammar*** *is a quadruple* $\mathbb{G} = \langle \Sigma, \Omega, \mathfrak{M}, R\rangle$ *where* $\langle \mathfrak{M}, R\rangle$ *is a boolean rewrite system and* $\Sigma, \Omega \in M$ *disjoint elements.*

This definition needs explanation. First, we can under some conditions on the grammar actually drop explicit mentioning of the terminal symbols. This can be useful technically. Notice also that the booleanness allows to have a single terminal symbol rather than a set on condition that this set is finite. But since we are dealing with a finite lexicon it necessarily is and we have therefore defined it this way. The notion of a labelled tree now needs to be adapted. Let *Ter* be the vocabulary (i. e. lexicon) and $\mathfrak{M}$ a mereology. A **partially labelled tree** over $\mathfrak{M}$ and *Ter* is a pair $\langle \mathbb{T}, \ell\rangle$ with $\ell : T \to M \cup Ter$. A **partial frame** over $\mathfrak{M}$ is a pair $\langle \mathbb{F}, \ell\rangle$ with $\ell : F \to M$. A partial frame arises from a partial tree by dropping all leaves, while a frame can be boosted up to a tree by adding a node under each leaf.

For a frame we can define a model relation with the grammar. We write $\langle \mathfrak{T}, x\rangle \models \mathsf{a}$ if $\ell(x) \sqsubseteq \mathsf{a}$. The reason we call this labelling *partial* is that it is no longer true that either $\langle \mathfrak{T}, x\rangle \models \mathsf{a}$ or $\langle \mathfrak{T}, x\rangle \models -\mathsf{a}$; thus we no longer have *bivalence*, that is, classical logic. Nevertheless, we *do* use classical logic, but the labels now express only part of what is or may be true of a node. A **(fully) labelled tree** is a partially labelled tree in which for every x the set

$$x^* = \{\mathsf{a} \in \mathfrak{M} | \langle \mathfrak{T}, x \models \mathsf{a}\}$$

is an ultrafilter in the mereology; this can only be if the lowest element in x^*, $\ell(x)$ is an atom of $\mathfrak{M}$.

As outlined above, we will not take labels as unanalysable. The labelling component itself will have the structure of a mereology. the algebra of labels is by definition meant for the non-terminals. A partially labelled tree is **generated** by $\mathbb{G}$ if it satisfies the *local condition* and the *boundary condition*. The **boundary condition** is that $\ell(r) \sqsubseteq \Sigma$ and that $\ell(x) \sqsubseteq \Omega$ iff x is a leaf. The local condition is that for an x which is not a leaf and has daughters $y_1, \dots, y_n$ the rule $\ell(x) \to \ell(y_1) \dots \ell(y_n)$ instantiates one of the rules of the grammar. In GPSG terms we say that $\ell(x) \to \ell(y_1) \dots \ell(y_n)$ have to satisfy the local admissibilty conditions set by the grammar. These conditions are given via the rules. Namely, let $\lambda = \mathsf{A} \to \mathsf{B}_1 \dots \mathsf{B}_m$ $\mu = \mathsf{K} \to \mathsf{L}_1 \dots \mathsf{L}_n$ be rules (alias local trees); then we write $\lambda \sqsubseteq \mu$ if $m = n$, $\mathsf{A} \sqsubseteq \mathsf{K}$ and $\mathsf{A}_i \sqsubseteq \mathsf{L}_i$ for all $i \leq n$. And then $\xi = \ell(x) \to \ell(y_1) \dots \ell(y_m)$ **instantiates** a rule ρ if $\xi \sqsubseteq \rho$. This concludes the definition of the local condition by a grammar. Intuitively, this makes sense as follows. Given a partially labelled tree $\langle \mathbb{T}, \ell \rangle$ and a node $x \in T$ we write $\ell(x) \sqsubseteq \mathsf{B}$ to say that in the labelled tree x is a node of **type** B. This is formally the case if in the mereology of labels $\mathfrak{L}$ $\ell(x) \sqsubseteq \mathsf{B}$. Now whenever we have a rule $\rho = \mathsf{A} \to \Gamma$ it is understood to mean that any node of type A immediately dominates a sequence $x^{\downarrow}$ of nodes of type Γ *whenever ρ is used to expand* A.

A boolean cfg is effectively a cfg. To this end one should note that a boolean rule $\rho = \mathsf{a} \to \Gamma$ effectively abbreviates a set of rules, namely all *precisifications* of ρ. To also be precise, σ is a **precisification** of ρ if $\sigma \sqsubseteq \rho$. Each tree in which a rule ρ is used at a point x instantiates a precisification of ρ in the sense that $\ell(x) \to \ell(x^{\downarrow})$ precisifies ρ. The local and the boundary conditions are translated as follows:

(*bd*) $\ell(r) \sqsubseteq \Sigma$
$\ell(x) \sqsubseteq \Omega$ if x is a leaf
(*lc*) $\ell(x) \to \ell(x^{\downarrow}) \sqsubseteq \rho$ for some $\rho \in R$

Since in a cfg all labels are mutually exclusive, we must take as labels of the grammar all maximal precisifications of $\mathfrak{M}$, in other words the **atoms** of $\mathfrak{M}$. Since $\mathfrak{M}$ is finite, it is atomic and so this construction goes through. Each element $\mathsf{b} \in M$ is uniquely determined by the set $\widehat{\mathsf{b}}$ of all atoms below it. Thus if $\langle \Sigma, \mathfrak{M}, R \rangle$ is a boolean cfg, let $\mathrm{At}(\mathfrak{M})$ be the set of atoms of $\mathfrak{M}$. Now replace a rule $\rho = \mathsf{a} \to \Gamma$ by the set of atomic precisifications of ρ

$$\rho^* = \{\mathsf{b} \to \Delta | \mathsf{b} \sqsubseteq \mathsf{a}, \Delta \sqsubseteq \Gamma, \mathsf{b} \in \mathrm{At}(\mathfrak{M}), \Delta \subseteq \mathrm{At}(\mathfrak{M})\}$$

and define $R^* = \bigcup \langle \rho^* | \rho \in R \rangle$. Then $\mathbb{G}^* = \langle \Sigma^*, \mathrm{At}(\mathfrak{M}), \Omega^*, R^* \rangle$ is a cfg with a *set* of start symbols Σ^*. The two grammars are equivalent in the sense that they admit the same trees with labels from $\mathrm{At}(\mathfrak{M})$. It is a standard procedure to reduce a cfg with a set of start symbols to another with a single start symbol. If we do not have a boolean cfg beacuse the mereology is infinite, we can produce an ordinary rule grammar via duality. This time, the rules are based on ultrafilters rather than atoms.

At this point we shall pause the development of the formal theory and reflect on the necessity to have infinite mereologies in the grammar. At first sight this seems totally unnecessary. But there are technical devices in nearly all grammars that practically amount to an infinite mereology. In HPSG the nodes are attributive value matrices, but in contrast to GPSG there is no a priori bound on their size. In GB elements are *indexed* and this also amounts to an infinite resource of symbols. In both grammar camps one might disagree. From the GB viewpoint the indices may be called extraneous, just a sort of helping device to keep track of what one is doing. That is, they might be good for the eye, but dispensable in principle. But this turns out to be a subtle matter. Indeed, some theories show that there is an a priori bound on the number of crossing extraction paths, and this implies that a finite stock of indices is sufficient. But some phenomena, which lead languages out of context freeness, require an analysis with an in principle unbounded stock of indices. Otherwise they cannot generate a non-context free language. However, this does not demonstrate that we need boolean indices rather than plain indices. Again, this is an empirical matter. But a proper formulation of the rules handling such indices needs to make reference to the set of indices that have hitherto been used, in case, for example, the new index needs to be strictly different. In order to characterize that suitably, we need to have finite unions and complements. And thus the index component of the grammar is boolean as well. It might be of some interest to speculate whether it consists of finite and cofinite sets of indices so that the indices are actually *there* in some sense like in $\mathfrak{Cof}$, or whether the indices are just given via descriptions as in $\mathfrak{Mod}$. (In fact, in the latter case the descriptions can with some right be said to be the indices.)

5.6 Presentation of Grammars and Memory Grammars

On the one hand boolean grammars just consist of a boolean algebra together with rules, on the other we prefer this boolean algebra to consist of specific terms such as *being an n-bar*, *being a v-zero trace* etc. From an abstract point of view, the grammar itself need not care about such an interpretation of its labelling discipline, so one and the same grammar can serve different purposes under different interpretations because we can think of the boolean labels in different ways. This means that what we should be dealing with are pairs $\langle \mathbb{G}, \epsilon \rangle$ where $\mathbb{G} = \langle \Sigma, \Omega, \mathfrak{M}, R \rangle$ is a boolean grammar and $\epsilon : F \to \mathfrak{M}$ a map from a finite set F of labels. This map can also be seen as a homomorphism $\overline{\epsilon}$ from $\mathfrak{Fr}(F)_o$ into the mereology $\mathfrak{M}$ of the boolean grammar. This homomorphism evaluates any term built fromthe symbols of F in the mereology – as long as it is not contradictory, in which case no value is given. The homomorphism is thus a partial map.

Theorem 5.6.1 *A* ***memory F-grammar*** *is a pair* $\mathbb{S} = \langle \mathbb{G}, \epsilon \rangle$*, where* $\mathbb{G} = \langle \Sigma, \Omega, \mathfrak{M}, R \rangle$ *is a boolean grammar and* $\epsilon : F \to M$ *a map.*

The concept of a *memory grammar* is rather natural if developed this way, but rather hard to justify intuitively. From an intuitive standpoint we expect the mereology of the grammar to be generated by the F-terms, that is, we expect to have access to all symbols of the mereology. But this is to say that $\overline{\epsilon}$ is surjective. If it is, we call the memory grammar **rational**. Non-rational grammars are nevertheless motivated by some considerations on the architecture of a linguistic theory. The leading idea behind them is that there might be classifications in the mereology of the grammar which are technically necessary to ensure the correctness of the grammatical output trees but from an intuitive syntactical viewpoint uninteresting. And that it might be undesirable that syntactic principles make reference to these distinctions. Such a distinction might be $\mathsf{VP}[-\mathsf{SLASH}\colon \mathsf{NP}]$ and $\mathsf{VP}[\mathsf{SLASH}\colon \mathsf{NP}]$ as opposed to just VP. [1] More clear, however, is the case of GB. Trees shown in GB books display very little information locally, opposite to GPSG and HPSG. The idea is that the missing local information can be recovered by the context, that is, the overall tree. In GB, the distinction just mentioned is never made in the labelling because the tree itself documents clearly whether or not we have an np-trace inside the vp at hand. If we aim at generating such trees with an ordinary boolean grammar we have a problem, because the global information that the labels do not display because it resides in the tree needs to be strictly locally accessible for a cfg.

Memory grammars solve this problem by making the global information locally accessible while denying outside access to the content of this information. This is just an optical trick. These grammars have the information to know how to proceed but we cannot spy into them to see what that information is. Nevertheless, we can analyze these grammars by artificially adding extra symbols to make the map ϵ surjective. These hidden states create what might suitably be called a *memory*. Even though we cannot see a difference by means of the F-symbols between these states, the grammar acts differently when being in one rather than the other. For the sake of concreteness let us assume that $F = \{\mathsf{F}_1\}$. There is no context free boolean grammar over $\mathfrak{Fr}(F)$ that allows to distribute F_1 on exactly those nodes of depth divisible by 3. This can only be achieved if we allow the grammar to internally count the number of successive $-\mathsf{F}_1$ on it's way down in the tree. In fact, there is a grammar based on three atoms that achieves this. We call them $\mathsf{A}_1, \mathsf{A}_2$ and B. Let us put $\Sigma = \mathsf{B}$, $\Omega = \{\mathsf{A}_1, \mathsf{A}_2, \mathsf{B}\}$ and

$$\begin{tikzcd}
\mathsf{B} \arrow[d, dash] & \mathsf{A}_1 \arrow[d, dash] & \mathsf{A}_2 \arrow[d, dash] \\
\mathsf{A}_1 & \mathsf{A}_2 & \mathsf{B}
\end{tikzcd}$$

Actually, we should be using sets of atoms rather than atoms, but this is marginal. Now we map $\epsilon(\mathsf{F}_1) = \{\mathsf{B}\}$. The distinction between A_1 and A_2 is purely induced by the need to keep track of the number of elapsed $-\mathsf{F}_1$. The number of features needed to achieve

[1] I deliberately said *might be*, because on a closer inspection the difference between the two might seem also relevant.

the counting can be measured quite effectively by comparing the image of $\overline{\epsilon}$ with the actual mereology the grammar uses itself. Or, we could simply ask how many more features we need to add to F in order to make ϵ surjective. In memory grammars the mereology splits into two parts; one part connected with the F-features and one memory of hidden part. Now if $\mathfrak{M} = \mathfrak{B}_o$ is the mereology of an F-grammar $\mathbb{S} = \langle \mathbb{G}, \epsilon \rangle$ then $\mathfrak{M}$ contains a subalgebra $im\overline{\epsilon}$ of all $\overline{\epsilon}$-images of boolean terms from F. So we have a surjective map $\mathfrak{Fr}(F) \twoheadrightarrow im\overline{\epsilon} \rightarrowtail \mathfrak{B}$. On the other hand there is a set $X \supseteq F$ such that a surjection $\kappa : \mathfrak{Fr}(X) \twoheadrightarrow \mathfrak{B}$ exists which extends ϵ. Put $M = X - F$. Then it follows that $\mathfrak{M} \cong \mathfrak{Fr}(F) \otimes \mathfrak{Fr}(M)/G$ for some filter G. The minimum cardinality of M is called the ***memory*** of $\mathbb{S}$ and is denoted by $m(\mathbb{S})$. In the context free case this can be defined via

$$m(\mathbb{S}) = \left\lceil {}^{2}log(\frac{\sharp(\mathbf{At}(\mathfrak{M}))}{\sharp(\mathbf{At}(im\epsilon))}) \right\rceil$$

$\mathbb{S}$ is ***rational*** or ***memory free*** if $m(\mathbb{S}) = 0$. In a boolean grammar, the number of generators is $\ulcorner\sharp(\mathbf{At}(\mathfrak{B}))\urcorner$, where $\sharp(\mathbf{At}(\mathfrak{B}))$ is the cardinality of the set of atoms and $\ulcorner r \urcorner$ for a real number r the least integer greater or equal to r.

It is known that context free languages can be parsed by a push-down automaton. This push-down automaton uses a stack of symbols from the grammar. Therefore, apart from the size of the stack in correspondence with a sentence to be parsed, the number of symbols manipulated by a grammar is an important complexity measure. In boolean grammars we have a boolean grammar as a basis. In the finite case we can simply consider it's set of atoms, that is, consider the corresponding ordinary cfg. But the problem is that in many cases the underlying boolean algebra is not finite, even though the symbols can be finitely presented. This case is too important to be ignored. An important number to monitor in boolean algebra is the *number of generators*. (Recall that a boolean algebra can be presented by a set of generators and relations.) In the infinite case this cannot be a number, but is a *cardinal number*. So, non-context free boolean grammars have at least $\aleph_0$ generators. Memory grammars give this classification an additional twist.

Memory grammars are not supposed to give away the exact labels. The mereological labels are not communicated; instead they give us a tree where the labels from F are positioned. So, a memory F-grammar generates trees with labels from $\mathfrak{Fr}(F)$, which we call *F-trees*. We write $\mathfrak{M} \models \mathsf{a} \sqsubseteq \mathsf{f}$ for a $\mathsf{f} \in \mathfrak{Fr}(F)$ and $\mathsf{a} \in M$ if $\overline{\epsilon}(\mathsf{f}) \sqsupseteq \mathsf{a}$. A memory F-grammar generates an F-tree $\mathfrak{T} = \langle T, \ell \rangle$ iff there is a tree $\mathfrak{U} = \langle T, \mu \rangle$ which the underlying boolean grammar generates such that $\mathfrak{M} \models \mu(x) \sqsubseteq \ell(x)$. So, secretly the memory grammar produces a tree with internal labels; these labels we can in most cases only partially access via the labels.

5.7 Cutting Grammars

Suppose that $\mathbb{G} = \langle \Sigma, \mathfrak{M}, R\rangle$ is a boolean grammar and $\zeta : \mathfrak{M} \to \mathfrak{N}$ a homomorphism of mereologies. Then we can define the image grammar $\mathbb{G}^\zeta$ of $\mathbb{G}$ under ζ as follows. Assume that $\rho = \mathsf{a} \to \mathsf{b}_1 \dots \mathsf{b}_n$ is a rule. Then two cases arise. The first is when ζ fails to be defined on either a or one of the b_i. In that case ρ^ζ is left undefined. In the other case we put $\rho^\zeta = \zeta(\mathsf{a}) \to \zeta(\mathsf{b}_1)\dots\zeta(\mathsf{b}_n)$. Finally, $R^\zeta = \{\rho^\zeta | \rho^\zeta \text{ is defined}\}$. Now $\mathbb{G}^\zeta = \langle \zeta\Sigma, \zeta\Omega, \mathfrak{N}, R^\zeta\rangle$, whenever all symbols are defined. In the same way we can define the image of a labelled tree under ζ. Namely, if $\mathfrak{T} = \langle \mathbb{T}, \ell\rangle$ is a labelled tree then if $\zeta(\ell(x))$ is defined on the whole tree, $\mathfrak{T}^\zeta$ is defined and equal to $\langle \mathfrak{T}, \zeta \circ \ell\rangle$.

Proposition 5.7.1 *Suppose $\mathfrak{T}^\zeta$ is defined. Then $\mathbb{G} \gg \mathfrak{T}$ implies $\mathbb{G}^\zeta \gg \mathfrak{T}^\zeta$.*

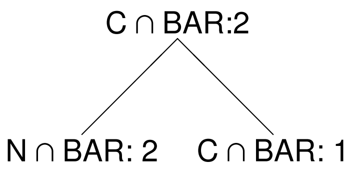
Proof. Σ^ζ is defined and by assumption we have $\ell(r) \sqsubseteq \Sigma$. Thus $(\zeta \circ \ell)(x) \sqsubseteq \zeta(\Sigma)$. Similarly for leaves. So much for the boundary condition. Now assume that x is not a leaf. and $\ell(x) \to \ell(y_1)\dots\ell(y_n) \sqsubseteq \rho$. We know that ζ is defined on $\ell(x)$ and all $\ell(y_i)$. Thus $(\zeta\circ\ell)(x) \to (\zeta\circ\ell)(y_1)\dots(\zeta\circ\ell)(y_n) \sqsubseteq \rho^\zeta$ and so $\ell(x) \to \ell(y_1)\dots\ell(y_n))^\zeta \sqsubseteq \rho^\zeta$. ⊣

By the representation theorems of § **??**, $\mathfrak{M} \cong \mathbf{2}^{\mathrm{At}(\mathfrak{M})}{}_o$ and $\mathfrak{N} \cong \mathbf{2}^{\mathrm{At}(\mathfrak{N})}{}_o$, and so the action of ζ can be studied by looking at the partial function it induces from $\mathbf{2}^{\mathrm{At}(\mathfrak{M})}{}_o$ to $\mathbf{2}^{\mathrm{At}(\mathfrak{N})}{}_o$; equivalently, we can study the corresponding boolean homomorphism. This homomorphism we denote by the same symbol. Since ζ is a homomorphism it is again enough to study the function $\widehat{\zeta} : \mathrm{At}(\mathfrak{M}) \to 2^{\mathrm{At}(\mathfrak{N})}$. Namely, every element a is the join of the atoms below it and thus $\zeta(\mathsf{a}) = \bigsqcup\langle \zeta(\mathsf{A}) | \mathsf{A} \sqsubseteq \mathsf{a}\rangle = \bigsqcup\langle \widehat{\zeta}(\mathsf{A}) | \mathsf{A} \sqsubseteq \mathsf{a}\rangle$. Any function $\iota : \mathrm{At}(\mathfrak{M}) \to 2^{\mathrm{At}(\mathfrak{N})}$ can be raised to a homomorphism provided it satisfies two things: $\iota(\mathsf{a}) \cap \iota(\mathsf{b}) = 0$ iff $\mathsf{a} \neq \mathsf{b}$ and $\bigcup\langle \iota(\mathsf{a}) | \mathsf{a} \in \mathrm{At}(\mathfrak{M})\rangle = \mathrm{At}(\mathfrak{N})$. To put it simple, ι must induce a partition on the atoms of $\mathfrak{N}$. Returning to ζ we can state that ζ is *surjective* if for all atoms $\sharp\widehat{\zeta}(\mathsf{A}) \leq 1$ and *injective* if $\sharp\widehat{\zeta}(\mathsf{A}) \geq 1$ for all atoms A. Surjections 'cut' atoms to scrap, namely those for which $\widehat{\zeta}(\mathsf{A}) = \emptyset$. The latter means that any atomic precisification of a rule which contains this atom will be banned from the image grammar. An injective ζ ambiguates symbols of the source grammar. If, namely, $\widehat{\zeta}(\mathsf{A})$ contains two atoms B and C then in any labelled tree generated by $\mathbb{G}^\zeta$ C and B are syntactically indistinguishable in the sense defined earlier. A grammar is *refined* if this cannot occur. We make this precise as follows. A map $\zeta : \mathfrak{M} \twoheadrightarrow \mathfrak{M}$ (surjective) is called a **refinement** if there exists an injection $\iota : \mathfrak{M} \rightarrowtail \mathfrak{M}$ such that for all fully labelled trees $\mathfrak{T}$ $\mathbb{G} \gg \mathfrak{T}$ iff $\mathbb{G}^{\zeta\circ\iota} \gg \mathfrak{T}$. $\mathbb{G}$ is **refined** if every refinement of $\mathbb{G}$ is bijective. To give a simple example, take the grammar $\mathbb{SET} = \langle 1, \mathfrak{Fr}(\mathsf{A}), R\rangle$ with R consisting of $1 \to 1$ and $1 \to 0$. It is clear that A and $-\mathsf{A}$ are absolutely equal and can be subsituted for each other anywhere in a tree. Another instance of reduction which is exemplified in grammar $\mathbb{L} = \langle \mathsf{A}, \mathfrak{Fr}(\mathsf{A}), R\rangle$ with R consisting of $\mathsf{A} \to \mathsf{A}$ and $\mathsf{A} \to 0$. It is clear that $-\mathsf{A}$ can never appear on any node of a generated tree, is not groundable. Thus the map $\zeta : \mathsf{A} \mapsto \mathsf{A}, -\mathsf{A} \mapsto 0$ simply cuts an atom that is never needed anyhow. In this case we cannot give an inverse ι as in the case of refinement, but we have $\mathbb{G} \gg \mathfrak{T}$ iff $\mathbb{G}^{\zeta} \gg \mathfrak{T}$.

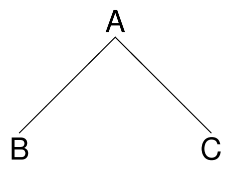
There is a one-to-one correspondency between images $\mathfrak{N}$ of surjections $\zeta : \mathfrak{M} \twoheadrightarrow \mathfrak{N}$ and filters, given by $F = h^{-1}(1)$. In effect, most surjections are given by a set of equations which can be turned into a filter F which in turn gives rise to a canonical homomorphism ζ_F. We write $\mathbb{G}/F$ for the grammar

5.8 Tensoring Grammars

A boolean grammar $\mathbb{G} = \langle \Sigma_{\mathbb{G}}, \Omega_{\mathbb{G}}, \mathfrak{M}, R\rangle$ can be understood as a complex consisting of a mereology regulating possible labellings and a rule component regulating the local admissibility. Such a view has been most fruitfully adapted in GPSG work. The rule



tells us that if a node branches into two daughters the tree is locally admitted at least if the mother is C∩BAR: 2, the left daughter is N∩BAR: 2 and the right daughter is C∩BAR: 1. We said *at least* because the rules concerning binary branching nodes specify a disjunctive condition on admissibility. Thus, a local tree



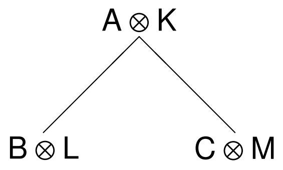
is admitted by a grammar if it is admitted by *at least one* of its binary branching rules.

We consider the set of rules to be decomposed into

$$R = \bigcup \langle R_n | n \leq k\rangle$$

where R_n collects all n-ary rules. Consider now a different grammar $\mathbb{H} = \langle \Sigma_{\mathbb{H}}, \Omega_{\mathbb{H}}, \mathfrak{N}, S\rangle$, with a similar decomposition of S into the sets S_n of n-ary rules. Assume the label algebras have nothing in common. This means that each grammar defines local admissibility using totally different features. The question to answered is how to define a grammar $\mathbb{G} \otimes \mathbb{H}$ such that for any labelled tree $\langle \mathbb{T}, \mu \otimes \nu\rangle$ we have $\mathbb{G} \otimes \mathbb{H} \gg \langle \mathbb{T}, \mu \otimes \nu\rangle$ iff $\mathbb{G} \gg \langle \mathbb{T}, \mu\rangle$ and $\mathbb{H} \gg \langle \mathbb{T}, \nu\rangle$. We have earlier discussed this in the context of cfgs under the heading of *pairing*. Pairing meant producing pairs of nonterminal symbols. By the correspondence of pairs of atoms with the tensor product of their set algebras the labelling algebra will be $\mathfrak{M} \otimes \mathfrak{N}$ and the start symbol $\Sigma_{\mathbb{G}} \otimes \Sigma_{\mathbb{H}}$. What about rules?

Assume a local tree of the following kind.



This tree is admitted by $\mathbb{G} \otimes \mathbb{H}$ if $\mathbb{G}$ admits $\mathsf{A} \to \mathsf{B} \quad \mathsf{C}$ and $\mathbb{H}$ admits $\mathsf{K} \to \mathsf{L} \quad \mathsf{M}$. This suggests the following definitions. Given two n-ary rules $\rho = \mathsf{A} \to \mathsf{B}_1 \dots \mathsf{B}_n$ and $\sigma = \mathsf{K} \to \mathsf{L}_1 \dots \mathsf{L}_n$ from $\mathbb{G}$ and $\mathbb{H}$ we put

$$\begin{array}{rcl} \rho \otimes \sigma & = & \mathsf{A} \otimes \mathsf{K} \to \mathsf{B}_1 \otimes \mathsf{L}_1 \dots \mathsf{B}_n \otimes \mathsf{L}_n \\ R_n \otimes S_n & = & \{\rho \otimes \sigma | \rho \in R_n, \sigma \in S_n\} \\ \mathbb{G} \otimes \mathbb{H} & = & \langle \Sigma_{\mathbb{G}} \otimes \Sigma_{\mathbb{H}}, \mathfrak{M} \otimes \mathfrak{N}, \bigcup_n R_n \otimes S_n\rangle \end{array}$$

Theorem 5.8.1 *Let $\langle \mathbb{T}, \mu \otimes \nu\rangle$ be a fully labelled tree where $\mu : T \to \mathfrak{M}$ and $\nu : T \to \mathfrak{N}$. Then*

$$\mathbb{G} \otimes \mathbb{H} \gg \langle \mathbb{T}, \mu \otimes \nu\rangle \textit{ iff } \mathbb{G} \gg \langle \mathbb{T}, \mu\rangle \textit{ and } \mathbb{H} \gg \langle \mathbb{T}, \nu\rangle.$$

Proof. We need to check the boundary condition and the local condition. Both are quite straightforward. One only needs to see that for rules $\rho_1, \rho_2, \sigma_1, \sigma_2$ we have $\rho_1 \otimes \sigma_1 \sqsubseteq \rho_2 \otimes \sigma_2$ iff both $\rho_1 \sqsubseteq \sigma_1$ and $\rho_2 \sqsubseteq \sigma_2$. ⊣

This theorem asserts that the tensor product has the desired properties. Finally, we need to deal appropriately with the case when grammars share labels. To combine grammars with shared labels in the correct way in fact we use the same rules but factor out unwanted assignments.

Definition 5.8.2 *The* ***tensor product*** *of* $\mathbb{G} = \langle \Sigma_{\mathbb{G}}, \Omega_{\mathbb{G}}, \mathfrak{M}, R\rangle$ *and* $\mathbb{H} = \langle \Sigma_{\mathbb{H}}, \Omega_{\mathfrak{H}}, \mathfrak{N}, S\rangle$ ***fibred*** *along a common submereology* $\mathfrak{L}$ *is denoted by* $\mathbb{G} \otimes_{\mathfrak{L}} \mathbb{H}$ *and defined by*

$$\mathbb{G} \otimes_{\mathfrak{L}} \mathbb{H} = \langle \Sigma_{\mathbb{G}} \otimes \Sigma_{\mathbb{H}}, \Omega_{\mathbb{G}} \otimes \Omega_{\mathbb{H}}, \mathfrak{M} \otimes_{\mathfrak{L}} \mathfrak{N}, \bigcup_n R_n \otimes S_n\rangle$$

We notice two extreme cases; the first is when $\mathfrak{L} = \mathbf{2}$. In this cases $\mathfrak{M} \otimes_{\mathfrak{L}} \mathfrak{N} = \mathfrak{M} \otimes \mathfrak{N}$ since $\mathbf{2}$ is always a subalgebra so that $\mathfrak{M}$ and $\mathfrak{N}$ cannot be said to share a non-trivial subalgebra. Consequently, $\mathbb{G} \otimes_{\mathfrak{L}} \mathbb{H} = \mathbb{G} \otimes \mathbb{H}$. The second extreme is when $\mathfrak{M} = \mathfrak{N} = \mathfrak{L}$. Then, as the labels of the two algebras $\mathfrak{M}$ and $\mathfrak{N}$ are identified and must be assigned in tandem it is clear that $\mathfrak{M} \otimes_{\mathfrak{L}} \mathfrak{N} \cong \mathfrak{M}$. Thus the local admissibility conditions of $\mathbb{G}$ and $\mathbb{H}$ are simply conjoined; $\mathbb{G} \otimes_{\mathfrak{N}} \mathbb{H} \gg \langle \mathbb{T}, \mu\rangle$ iff both $\mathbb{G} \gg \langle \mathbb{T}, \mu\rangle$ and $\mathbb{H} \gg \langle \mathbb{T}, \mu\rangle$.

Another interesting construction is the *amalgamation*. The amalgamation of two grammars unifies the tensor product and the fibred tensor product.

Definition 5.8.3 *Let* $\mathbb{G}$ *and* $\mathbb{H}$ *be boolean grammars and* F *a thin filter. Then* $(\mathbb{G} \otimes \mathbb{H})/F$ *is called an* ***amalgamation*** *of* $\mathbb{G}$ *and* $\mathbb{H}$*.*

Analogous operations can now be defined on memory F-grammars. A first guess, that if $\epsilon : F \to \mathfrak{M}$, $\zeta : F \to \mathfrak{N}$, then the map $\epsilon \otimes \zeta : F \to \mathfrak{M} \otimes \mathfrak{N}$ is a map from F into the underlying mereology of the tensor product, is incorrect. Namely, it is a strict condition that the labels must mean the same in both grammars, so we need to identify their interpretation. This is the generic case of a fibred tensor product. The appropriate labelling mereology in the tensor F-grammar is not the tensor product but $\mathfrak{M} \otimes_{\mathfrak{Fr}(F)} \mathfrak{N}$. Thus with $\langle \mathbb{G}, \epsilon\rangle$ and $\langle \mathbb{H}, \zeta\rangle$ F-grammars, the tensor product is defined as $\langle \epsilon\zeta, \mathfrak{M} \otimes_{\mathfrak{Fr}(F)} \mathfrak{N}\rangle$.

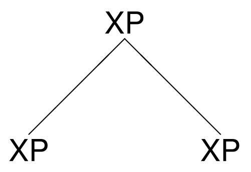
5.9 Examples and Discussion

As an example to illustrate boolean grammars we present (one version of) $\overline{X}$-syntax. For the sake of concreteness let us take $\overline{X}$-syntax over two labels, S and N; it has in addition to these two also the labels BAR:0, BAR:1 and BAR:2. Not only is N inconsistent with S, the bar levels are also exclusive. This allows 6 consistent atoms, namely, N ∩ BAR:0, N ∩ BAR:1, N ∩ BAR:2, S ∩ BAR:0, S ∩ BAR:1, S ∩ BAR:2. The start symbol is S ∩ BAR: 2. In addition to this, the following rules are rules of $\overline{X}$-syntax. (We are assuming a head-final language here, we allow for recursion on non-branching symbols and we also exclude pruning ($\overline{\overline{X}} \to X$ and zero level adjunction. This is of course only for the purpose of illustration.) We write $\overline{\mathbb{X}}$(S,N) for this grammar.

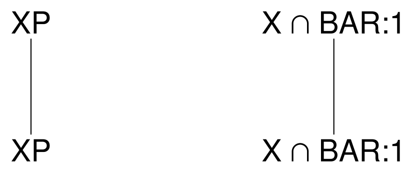
$$\begin{array}{lcll}
\mathsf{S} \cap \mathsf{BAR{:}2} & \to & \mathsf{S} \cap (\mathsf{BAR{:}2} \cup \mathsf{BAR{:}1}) & \\
\mathsf{S} \cap \mathsf{BAR{:}2} & \to & \mathsf{BAR{:}2} & \mathsf{S} \cap (\mathsf{BAR{:}2} \cup \mathsf{BAR{:}1}) \\
\mathsf{S} \cap \mathsf{BAR{:}1} & \to & \mathsf{S} \cap (\mathsf{BAR{:}1} \cup \mathsf{BAR{:}0}) & \\
\mathsf{S} \cap \mathsf{BAR{:}1} & \to & \mathsf{BAR{:}2} & \mathsf{S} \cap (\mathsf{BAR{:}1} \cup \mathsf{BAR{:}0}) \\
\mathsf{N} \cap \mathsf{BAR{:}2} & \to & \mathsf{N} \cap (\mathsf{BAR{:}2} \cup \mathsf{BAR{:}1}) & \\
\mathsf{N} \cap \mathsf{BAR{:}2} & \to & \mathsf{BAR{:}2} & \mathsf{N} \cap (\mathsf{BAR{:}2} \cup \mathsf{BAR{:}1}) \\
\mathsf{N} \cap \mathsf{BAR{:}1} & \to & \mathsf{N} \cap (\mathsf{BAR{:}1} \cup \mathsf{BAR{:}0}) & \\
\mathsf{N} \cap \mathsf{BAR{:}1} & \to & \mathsf{BAR{:}2} & \mathsf{N} \cap (\mathsf{BAR{:}1} \cup \mathsf{BAR{:}0}) \\
\mathsf{BAR{:}0} & \to & 0 &
\end{array}$$

This amounts to a total of 25 rules in an ordinary cfg. This grammar is indeed a normal grammar. Notice that the algebra of labels is a tensor product of the algebras $\mathfrak{Bar}$ and the algebra of the categories. The former is isomorphic to $\mathbf{2}^3$ the latter isomorphic to $\mathbf{2}^2$; so the resulting algebra is isomorphic to $\mathbf{2}^6$.

Quibble. Is a rule such as



a rule of $\overline{X}$-syntax? Normally, phrasal categories are not considered recursive, that is, there are no phrasal adjuncts. But this holds mostly only of deep structure. In (Chomsky, 1986a), there are phrasal adjuncts at surface structure, so the question does arise here. Real gbers would actually say that $\overline{X}$-syntax does only spell out deep structure trees. But this only defers the problem. Suppose we have a surface structure tree, how do we recognize in this structure which is the adjunct – the left or the right XP? In the sequel we will ban (even in the surface structure) such productions together with empty productions such as



We can separate in (boolean) grammars two things: a structure building component and a component spreading (boolean) labels over the structure. In boolean grammars, this spreading of labels may itself be viewed as a spreading of the features of which the labels are made. Recall namely that if $\langle \mathbb{T}, \ell \rangle$ is a fully labelled tree, the labels $\ell(x)$ are atoms and therefore rectangles. If $\ell : T \to \mathfrak{Fr}(n)$ then ℓ decomposes into assignments $\mu_i : T \to \mathfrak{Fr}(1)$ concerned with the feature F_i. And the latter are completely determined by the sets $[[\mathsf{F}_i]] = \{x | \langle \mathbb{T}, \ell, x \rangle \models \mathsf{F}_i\}$. This is, I guess, also intuitively clear; given that features are positively or negatively assigned to every node of the tree it suffices to give all the sets $[[\mathsf{F}_i]]$ in order to create the labelling ℓ. We wish to think of the grammar as actively assigning values at the nodes; that causes some problems of identity if we assume that labels can be partial. But we want to think of these partial objects rather as abbreviations of the set or their precisifications. So the grammar actively produces fully labelled trees, we are just too sloppy to write down which one we mean. Nevertheless, this does not mean that we are back to plain cfgs because the labels are stills assumed to have internal structure. The precise nature of the labels is clarified exactly by the labelling algebra. Here we also want to stipulate that it is always in it's incarnation as a presented algebra that we want to look at it. This presentation uses a finite set $\mathbb{F}_k \subset \mathbb{F} = \{F_i | i \in \omega\}$. So, from the (infinite) stock of features that we can possibly use, the grammar actively distributes only a finite number of them. These features, however, are treated as logical constants; all other features that are not assigned in this grammar are not simply undertermined and can be precisified at random. The number of labels that are distributed by the grammar is therefore unique because it depends on the presentation of te labelling algebra. If the number is k we say that the grammar is a $k^\sharp$-grammar.

If we have no labels at all, that is, if we have an $0^\sharp$-grammar, then this grammar generates just the bare trees, or, as we will say, the *slots* into which labels can be inserted by tensoring with other boolean grammars Therefore $0^\sharp$-grammars will also be called **slot grammars**. Slot grammars present the structural aspect of grammars without the labellings. The structure component of a grammar is completely determined by its *similarity type*. This type is a set of natural numbers. We put $\mathrm{tp}(\mathbb{G}) = \{n | (\exists \rho \in R)(\rho = \mathsf{A} \to \mathsf{B}_1 \ldots \mathsf{B}_n)\}$. In other words, $\mathrm{tp}(\mathbb{G})$ collects all branching numbers for rules of $\mathbb{G}$. For natural languages the types are normally assumed to be $\{1, 2\}$. We call $\exp(\mathbb{G}) := \mathit{max}\ \mathrm{tp}(\mathbb{G})$ the **expansivity** of $\mathbb{G}$. It is worthwile noting that $\mathrm{tp}(\mathbb{G} \otimes \mathbb{H}) \subseteq \mathrm{tp}(\mathbb{G}) \cap \mathrm{tp}(\mathbb{H})$ and so $\exp(\mathbb{G}) \leq \mathit{min}\{\exp(\mathbb{G}), \exp(\mathbb{H})\}$. Equality need not hold for example with $\mathbb{G}$ consisting of the rules $1 \to 1 \quad 0, 1 \to 0$ and $\mathbb{H}$ consisting of the rules $1 \to 0 \quad 1, 1 \to 0$. $\Sigma_{\mathbb{G}} = \Sigma_{\mathbb{H}} = 1$. Then $\mathbb{G} \otimes \mathbb{H}$ has type $\{1\}$ while both $\mathbb{G}$ and $\mathbb{H}$ have type $\{1, 2\}$. A slot grammar is uniquely determined by its type.

5.10 Observables and Unobservables

We have seen that the mereology of a memory grammar can be seen as labelled throughout. This is convenient for practical calculations. It is very difficult to be consistent in using features only when they stem from the set F and not access the internal mereology with a full set of features. Thus it is a good fiction to consider the mereology to be presented as generated by a set X of labels with a set of equations, or as a homomorphism $\kappa : \mathfrak{Fr}(X)_o \to \mathfrak{M}$. The set X naturally splits into two sets, F and it's complement. We call a feature $\mathsf{O} \in X$ an **observable** if it is in F and **unobservable** if not in F. This terminology is chosen with hindsight. We are not proposing that the split into observables and unobservables is arbitrary. To the contrary, we want to argue that it is better to think of the observables as the features that can be fixed by looking at the outer form of language, i. e. as language presents itself to us directly via the strings, while the unobservables are introduced step by step in the process of *coding* certain syntactic principles into the grammar. These principles are of course also necessitated by the facts, but in a less direct way, and are invariably dependent on the language with which we allow to talk about principles.

We will therefore propose that classification via observables must reflect *morphological* difference. An example will make this clear. The uninflected verb *borrow* as it occurs in *Can I borrow this book?* cannot be distinguished materially from the verb phrase *(borrow t)* as in *This is the book that I used to borrow.* There has been a long debate about this subject, to be correct, but the outcome in favour of a materially attestable basis for traces is quite meager. I admit, though, that what counts as material manifestation is debatable. First of all, it depends on whether we take language as written, or language as spoken; second, it depends on whether we take phonology (even semantics) into account. These questions are, quite clearly, not to be solved by a theoretical debate but by empirical study. We can despite all this reach the following consensus. We suitably restrict – though only for the purpose of exposition – the attention to written English (occasionally other written languages). In that case the material data is the string as it occurs in the written sentence. Clearly, traces leave no trace in a written sentence, and this means that the difference between a verb and a verb followed by a trace is unattestable just by looking at the strings. Let us take, then, a feature O and ask whether the difference between O and $-\mathsf{O}$ is significant in this sense. Here, we need the lexicon. The lexicon allows to define a map $^\flat : \mathfrak{M} \to 2^{V^*}$, where V is the vocabulary. This map transports a grammar symbol a into the set of all strings that can be classified or analysed as constituents of type a. With respect to this evaluation map we can state a principle of what to choose as observable.

Principle 5.10.1 (Observability) *A feature O is to be classified as* **observable** *if only $\mathsf{O}^\flat \neq (-\mathsf{O})^\flat$.*

This is only a mild form of observability because it constrains the grammar only with respect to the features that have been chosen from the outset. So it does *not* tell us which features to select as critical (this must be done elsewhere) but it does tell us which of them is to be seen as an observable feature. However weak, it does have consequences that cannot be overlooked. With nearness restrictions yet to be defined, let us remark that any such condition that will be coded into a grammar can refer only to observables. Most grammatical conditions already comply with this restriction, but the whole binding theory has to be revised in this light. If namely indices are unobservable, then binding conditions and the *i-over-i-principle* can simply not be expressed in their usual form. Two protests that may be voiced against this shall be discussed. First, indices are seen as an integral part in syntax because they do have a reflex in semantics. They express distinctness and non-distinctess. I have a two-fold answer to that. First, this claim means to give up the autonomy of syntax at least partly; in any case it means to concede that LF is more than a syntactic symbolism. Secondly, it is quite conceivable to count semantics as raw data (though I would prefer not to here) and to let $^\flat$ map classifications into annotated strings so that the difference *is* attestable. This is possible.

A second objection would be that indices are necessary to ensure the correctness of the antecedent-trace relation. I will say more about this later. It is one of the points of this whole book to demonstrate that indices are eliminable, and that they must be eliminated if GB wants to get clear about the meaning of it's own symbolism. It is simply not true that indices generally have a semantic interpretation. Linguists in GB are concerned with getting distributional facts right at all costs, and indices (in connection with movement) are intuitively useful. But they are a rather dangerous device simply because they are unobservable. At an initial stage one might use them, but the aim should be to get rid of them as soon as possible.

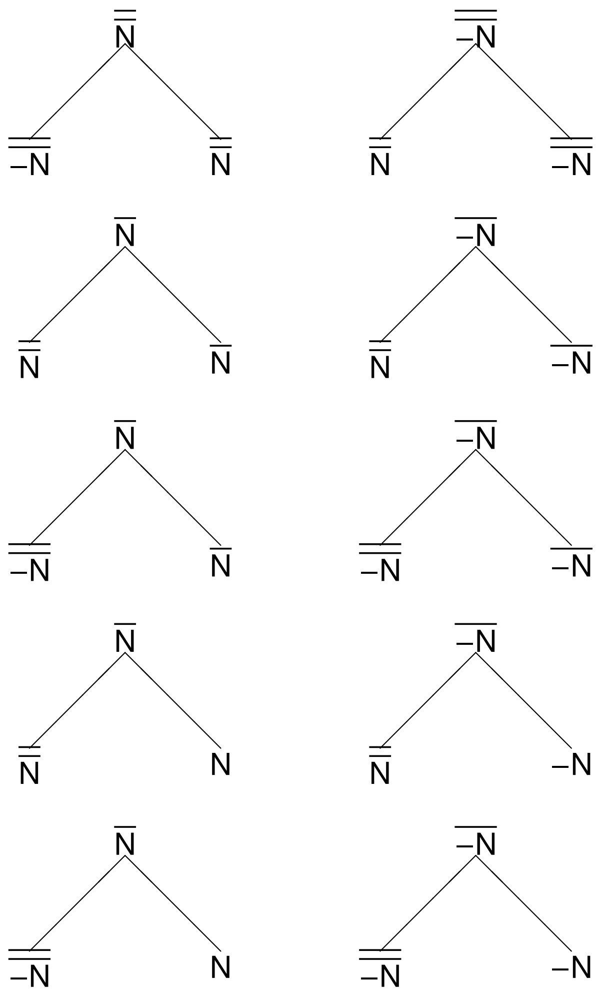
5.11 Modularity

Modularity has become an important criterion in linguistic ideology. It is desired that a syntactic theory and perhaps any linguistic theory be modular by which is meant that it is explicitly decomposed into subtheories that are inherently simple and whose interaction produces the complexity of the entire theory. We said *is decomposed* and not *can be decomposed* because modularity is a property of a theory, that is, something that is man made and does not decompose automatically. This subtle point might be the point of argument in many debates of whether such and such theory is modular. Even though a theory might be modularized it need not be modular; this is a question of presentation not of content. Of course, modularization can be understood formally as a procedure to modularize a theory in such a way that each restriction is associated with its own subtheory; such a modularization will be empirically empty. We do want the subtheories to be empirically motivated such that their independent existence is attestable.

In the context of boolean grammars modularization is the process of decomposing grammars into reduced tensor grammars. In the best possible case a grammar completely decomposes into grammars of type $\mathbf{1}^\sharp$ which contain rules governing one single feature. This, however, is in most cases impossible. Try e. g. decomposing $\overline{\mathbb{X}}(\mathsf{S}, \mathsf{N})$ in this way. But intermediate solutions are possible and yield satisfactory results.

EXAMPLE. $\overline{\text{X}}$-syntax uses an algebra of labels which can be decomposed into an algebra $\mathfrak{L}$

of levels and an algebra $\mathfrak{B}$ of basic syntactic categories. We have described it earlier. If we consider $\mathfrak{B}$ being based on a single feature, say N, the rules are the following, assuming a head-final language:



$\overline{\overline{\mathsf{N}}}$
$\overline{\overline{-\mathsf{N}}}$
$\overline{\mathsf{N}}$
$\overline{\overline{-\mathsf{N}}}$
$\overline{\mathsf{N}}$
$\overline{\overline{-\mathsf{N}}}$
$\overline{\mathsf{N}}$
$\overline{\overline{\mathsf{N}}}$
$\overline{\mathsf{N}}$
$\overline{-\mathsf{N}}$
$\overline{\overline{\mathsf{N}}}$
$\overline{-\mathsf{N}}$
$\overline{\mathsf{N}}$
$\overline{\overline{-\mathsf{N}}}$
$\overline{\mathsf{N}}$
$\overline{-\mathsf{N}}$
$\overline{\overline{-\mathsf{N}}}$
$\overline{-\mathsf{N}}$
$\overline{\mathsf{N}}$
$\overline{\overline{\mathsf{N}}}$
N
$\overline{-\mathsf{N}}$
$\overline{\overline{\mathsf{N}}}$
$-\mathsf{N}$
$\overline{\mathsf{N}}$
$\overline{\overline{-\mathsf{N}}}$
N
$\overline{-\mathsf{N}}$
$\overline{\overline{-\mathsf{N}}}$
$-\mathsf{N}$

This grammar looks exactly the same for all other symbols. Thus, $\overline{\mathbb{X}}(\mathsf{S})$ is isomorphic to $\overline{\mathbb{X}}(\mathsf{N})$. We get $\overline{\mathbb{X}}(\mathsf{S}, \mathsf{N})$ by tensoring the corresponding grammars identifying the level algebra $\mathfrak{Bar}$ and forcing the equation $\mathsf{S} = -\mathsf{N}$.

$$\overline{\mathbb{X}}(\mathsf{S}, \mathsf{N}) = (\overline{\mathbb{X}}(\mathsf{S}) \otimes_{\mathfrak{Bar}} \overline{\mathbb{X}}(\mathsf{N}))/\{\mathsf{S} = -\mathsf{N}\}$$

This is not hard to check. Any other $\overline{\text{X}}$-syntax decomposes into one-feature grammars; a possible exception is that some times there are no coassignment restrictions. For example, if V and INFL are coassignable then

$$\overline{\mathbb{X}}(\mathsf{V}, \mathsf{INFL}) = \overline{\mathbb{X}}(\mathsf{V}) \otimes_{\mathfrak{Bar}} \overline{\mathbb{X}}(\mathsf{INFL})$$

Proceeding in an analogous way with other grammars it is possible to decompose large grammars into manageable components that are fused by reduced tensor products. This defines a modularization of boolean grammar. The degree of modularization is measurable with the number of the factors and the size of the reducing filters in comparison with the grammar itself. The more components the better, the larger the filters by which to reduce the worse.

Chapter 6

Modal Logic and Syntax

6.1 A Brief Introduction to Modal Logic

The story of name forming operators was quite complex; however, it emerged that they can (in the really non-trivial cases) be shown to arise from an underlying relational structure on the tree. This makes modal logic the appropriate tool for analysis. Before we can do so we need to learn about it. The standard approach is via frames. A **frame** is a set ordered by a binary relation; hence, a frame is a pair $\mathfrak{f} = \langle f, \lhd \rangle$ where f is a set, called the **set of possible worlds** and $\lhd \subseteq f \times f$ a binary relation on f called the **accessibility relation**. Frames are often called **kripke-**frames after Kripke who was one of the first to use them in modal logic. For our purposes it is quite helpful to think of f as a set of locations and of $\lhd$ as a reachability or vicinity relation. If $x \lhd y$ then y is declared to be **near** to x or **accessible** from x. We also speak of y as a **successor** of x. Ordinary propositions can assume different values on different locations. For example, if f consists of the sets of rooms of my flat and p represents the proposition *I am here* then p can be true of the kitchen while being false for the living room and vice versa. This is a typical behaviour of indexicals such as *here*. Or, if q represents the proposition *has windows* it can be true both of the kitchen and the living room, to give an example where the truth at one place does not exclude the truth of the other. The accessibility relation is given by the system of doors. If there is a door from the kitchen to the living room, then the living room is *near* (in the technical sense) to the kitchen. Notice that this relation is symmetric, so that we an conclude that the kitchen must also be near to the living room. We can easily think of asymmetrical relations if we take f to be the set of road crossings of Paris and specify $x \lhd y$ if x is directly connected with y and one is allowed to drive from x to y. The numerous one-way roads make this quite an interesting frame.

The notion of propositions being true at worlds or locations is formalized by a function β that gives for each propositional variable or constant p a set $\beta(p) \subseteq f$; formally, β :

$Var \to 2^f$. We call $\langle \mathfrak{f}, \beta\rangle$ where $\mathfrak{f} = \langle f, \lhd\rangle$ a **kripke-model**. Given a point $x \in f$ we write $\langle \mathfrak{f}, \beta, x\rangle \models p$ in case $x \in \beta(p)$ and we write $\langle \mathfrak{f}, \beta, x\rangle \not\models p$ otherwise. In the first case we say that x is **true** at x in the model and in the second case we say that p is **false**. We can boost this up to more complex expressions by saying that $p \cap q$ is true at x iff both p and q are true at x, that $p \cup q$ is true at x iff either p or q are true at x, and that $-p$ is true at x iff p is false. Then all laws of boolean logic hold for the concept *being true at x*, regardless of x and the kripke-model it lives in. So far so good. Now we introduce the possibility to express facts that hold not at the location or world itself but which hold at another one. At this point the accessibility relation begins to play an active part. We introduce two new symbols, $\Box$ and $\Diamond$. Both are unary operators on propositions and form new propositions. So if p is a proposition, so are $\Box p$ and $\Diamond p$. They are read *it is necessary that p* and *it is possible that p*. The crucial fact is that *is possible* is interpreted as *there is a nearby location* or *there is an accessible world* while *is necessary* is interpreted *for all nearby locations* or *for all accessible worlds*. Formally, this looks as follows.

$$\begin{array}{lll} \langle \mathfrak{f}, \beta, x\rangle \models \Diamond P & \text{iff} & \text{there is a } y \text{ such that } x \lhd y \text{ and } \langle \mathfrak{f}, \beta, y\rangle \models P \\ \langle \mathfrak{f}, \beta, x\rangle \models \Box P & \text{iff} & \text{for all } y \text{ such that } x \lhd y \text{ holds } \langle \mathfrak{f}, \beta, y\rangle \models P \end{array}$$

Let's return to Paris. Suppose we are at some crossing x with our car. Let p be the proposition *there is a traffic jam*. If p holds at x already, we are stuck. If not, we are still not free of trouble. Suppose, namely that there are three directions to go (say, going north is forbidden at the moment because we would be entering a one way road) and that at all the next crosspoints there is a traffic jam. Then it holds at x that $\Box p$! An informal rendering would be *everywhere we may go, there is a traffic jam*. Equivalently, $-\Diamond - p$ is true; this amounts to saying that *nowhere we can escape a traffic jam*. Indeed, one can check that $-\Diamond - p \equiv \Box p$ and that $\Diamond p \equiv -\Box - p$. The assertion $\Diamond - p$, by the way, tells us that there is at least one direction free of traffic jam. If we want to get out of Paris, $\Diamond - p$ is still not enough because $\Box\Box p$ may still hold. We see that modal logic is in effect a language to talk about facts in different locations. Though it is a rather impoverished language we can make it quite powerful by assuming more than one nearness or accessibility relation. Basically, we can introduce a new relation $\blacktriangleleft$ on the road crossings of Paris where $x \blacktriangleleft y$ exactly if we as pedestrians can go from x to y. Obviously, in the inner city of Paris every road connection by car is a road connection by foot so $\lhd \subseteq \blacktriangleleft$. We can introduce new propositional operators $\blacklozenge$ and $\blacksquare$ which again mean *somewhere nearby it holds that* and *everywhere nearby it holds that* but now nearness is calculated with respect to $\blacktriangleleft$. Hence the following truth assignment rules hold.

$$\begin{array}{lll} \langle \mathfrak{f}, \beta, x\rangle \models \blacklozenge P & \text{iff} & \text{there is a } y \text{ such that } x \blacktriangleleft y \text{ and } \langle \mathfrak{f}, \beta, y\rangle \models P \\ \langle \mathfrak{f}, \beta, x\rangle \models \blacksquare P & \text{iff} & \text{for all } y \text{ such that } x \blacktriangleleft y \text{ holds } \langle \mathfrak{f}, \beta, y\rangle \models P \end{array}$$

It is clear that we have endless possibilities of introducing nearness relations and corresponding operators and we will shortly see that syntactic domains allow for a variety of nearness domains to be captured not by one but by a (limited) number of modal operators. Modal logic is by and large the study of the logic that result by taking a set of kripke-frames and studying the set of all propositions and rules that are true on all of them. The simplest case is when we take simply the set of all kripke-frames. For the moment we will also assume just one operator. Then the following are axioms.

PC.	All classical tautologies.
BD.	$\Box(p \to q). \to .\Box p \to \Box q$
DU.	$\Diamond p. \leftrightarrow . - \Box - p$

Furthermore we have the rule of Modus Ponens (from $p \to q$ and p infer q) and the Modus of Necessitation (if p is a theorem, $\Box p$ is a theorem as well). This presents the basic system **K** (again named after Kripke). It is possible to derive the following theorems.

0-homomorphy.	$\Diamond 0. \leftrightarrow .0$
$\cup$-distribution.	$\Diamond(p \cup q). \leftrightarrow .\Diamond p \cup \Diamond q$
$\cap$-distribution.	$\Box(p \cap q). \leftrightarrow .\Box p \cap \Box q$

It is not hard to verify that on all kripke-models these equivalences hold in all worlds. If we study not all kripke-frames but only a proper set $\mathfrak{F}$ of them we derive a logic $L\mathfrak{F}$ of formulas that hold on all models based on kripke-frames from $\mathfrak{F}$:

$$L\mathfrak{F} = \{P|(\forall \mathfrak{f} \in \mathfrak{f})(\forall \beta : Var \to 2^f)(\forall x \in f)(\langle \mathfrak{f}, \beta, x\rangle \models P)\}$$

$L\mathfrak{F}$ turns out to be an extension of **K** by some set of extra axioms. If $\mathfrak{F}$ is the class of frames satisfying certain nice properties we an actually characterize the frames of $\mathfrak{F}$ by some modal axioms. For example, on all reflexive frames, that is, on all frames $\langle f, \lhd\rangle$ where $x \lhd x$ for all x, we have that $p \to \Diamond p$ is true regardless of the valuation and the point chosen. Namely, if p holds at x then $x \lhd x$ and so by the truth definition $\Diamond p$. For example, let $\lhd$ be the relation of *being within 1 km distance*. This relation is reflexive. For matters of concreteness, take p to be *there is a tree*. Then clearly, if we have a tree here we also have a tree somwhere within 1 km. The axiom $p \to \Box\Diamond p$ characterizes *symmetric* relations. Take my flat again; if I am in this room, then no matter through which door I pass, I can go back to this room through at least one door. To conclude this discussion we point out that the Paris road system satisfies the axiom $\Diamond p \to \blacklozenge p$ or *whatever happens at a crosspoint to which we may drive happens at a crosspoint that one can go to*. This is a direct logical equivalent of the fact that $\lhd \subseteq \blacktriangleleft$.

Regarding our intended application there are some quite special logics, namely $\mathbf{Alt}_1$, **D**, $\mathbf{D}.\mathbf{Alt}_1$ and **Triv**. $\mathbf{Alt}_1$ is the logic of frames in which every point x has at most one accessible point, i.e. $x \lhd y, x \lhd z$ imples $y = z$. **D** is the logic of frames in which every point has at least one successor. $\mathbf{D}.\mathbf{Alt}_1$ is the logic of the frames in which every point has exactly one accessible point. And **Triv** is the logic of the frame $\mathfrak{f} = \langle\{x\}, \{\langle x, x\rangle\}\rangle$, the frame consisting of exactly one point, which is reflexive. The logics can be characterized by the following axioms. We will not show here that this is so.

Alt$_1$.	$\Diamond p. \to .\Box p$
D.	$\Diamond 1$
Triv.	$\Diamond p. \leftrightarrow .p$
D.Alt$_1$.	$\Diamond p. \leftrightarrow .\Box p$

Let us briefly mention the some connections with AVMS. Attribute-value formalisms are extremely powerful. In principle, any term can be coded into AVMS. Moreover, HPSG benefits from the possibility to code lists and trees into AVMS. That the latter is possible should not be surprising. The operators $\Diamond$, $\Diamond$, $\Diamond$ of the next section satisfy **Alt**$_1$ and can thus be read as attributes. This allows to code trees bottom-to-top. But a coding top-to-bottom is possible with the help of $\Diamond_\ell$ which relates a node with its left-most daughter. Then just $\Diamond_\ell$ and $\Diamond$ suffice to describe fully the structure of a given tree with an AVM. In HPSG this is used to realize information exchange between items in a syntactic tree; one can namely state via this coding that e. g. subject and verb agree by stating a so-called *reentrancy condition* or *path equation*,

$$\textsc{sub} : \textsc{agr} \approx \textsc{vp} : \textsc{agr}$$

In modal logic this can be axiomatically expressed by the postulate

$$\langle\textsc{sub}\rangle\langle\textsc{agr}\rangle p. \leftrightarrow .\langle\textsc{vp}\rangle\langle\textsc{agr}\rangle p$$

This statement of agreement between a verb phrase and a subject is parasitic on the phrase structure rules, that is, the grammar, in which these constraints are supposed to be implemented. The modal axiom states quite directly what the intention is, namely, the exchange of information between two nodes in a tree. In definite clause grammars we have boolean variables, and so the path equation can be coded directly into rules. The danger, however, is that this direct implementation blurs the distinction between finite and infinite AVMS. A priori the AVMS for HPSG can be infinite – as opposed to GPSG – and so it is not clear that the path equation lays a connection that allows a transfer of finite information between nodes. The latter is important. If only a finite amount of information is exchanged, it can be implemented into a boolean cfg without the use of variables for structures, by a simple case by case analysis. If the amount of information is infinite, boolean cfgs cannot guarantee the proper exchange of this information. This means, however, that implementing this exchange principle means transcending context freeness.

6.2 Propositional Dynamic Logic

We have just seen that we can have as many relations on a given set as we want and can connect a modal operator with it; moreover, if the two relations are interwtined in some non-trivial way this will be reflected by an axiom that the frame satisfies. Instead of keeping a fixed number of operators and relations we can exploit the aspect that binary relations can be combined by certain operations such as intersection, union and composition. Dynamic logic concentrates on this aspect. In dynamic logic we start with a basic set of abstract relations (called *programs*) and a basic set of propositions. We can combine propositions in the usual way; however, we can also combine abstract relations via union, concatenation and the Kleene-star. (Intersection proves to be rather badly behaved in modal terms so is generally left out of consideration.) Moreover, for every proposition P the expression $P?$ (read: *P test*) is an abstract relation. The concrete details are as follows. The vocabulary consists in

- A set $P_0 = \{\pi_1, \ldots, \pi_n\}$ of programs
- A set V_0 of propositional variables
- The program connectives $\cup, ;, ^*$
- The propositional connectives $\neg, \wedge, \rightarrow$ etc.
- The test ?
- The modality formation operators $[-], \langle - \rangle$

It is just for convenience that we assume P_0 to be finite. Well-formed expressions are defined thus. We have two types of well-formed expressions, namely propositions and programs. Propositions are built the usual way from propositions using the boolean connectives. Moreover, if π, σ are programs, then $\pi \cup \sigma$, $\pi; \sigma$ and π^* are also programs. If P is a proposition, $P?$ is a program and if π is a program and P a proposition then $[\pi]P$ as well as $\langle \pi \rangle P$ are propositions. A *dynamic frame* is a pair $\mathfrak{f} = \langle f, R_1, \ldots, R_n \rangle$. It is clear that the R_i serve as the relations that instantiate the programs in this dynamic frame. In dynamic logic one thinks of f as the states of a machine (say, a computer) and of the R_i as the basic operations that this computer performs. A model based on this dynamic frame is a pair $\langle \mathfrak{f}, \beta \rangle$ where $\beta : V_0 \rightarrow 2^f$ is a valuation on the propositions. The clauses for acceptance at a world are listed below. In formulating them we have used the convention π^n. This is defined inductively by $\pi^0 = \{\langle x, x \rangle | x \in f\}$, $\pi^{n+1} = \pi; \pi^n$. So, π^n is the n-fold iteration of π.

$\langle \mathfrak{f}, \beta, x \rangle \models p$	iff	$x \in \beta(p)$
$\langle \mathfrak{f}, \beta, x \rangle \models \langle \pi \rangle P$	iff	exists y such that $x \pi y$ and $\langle \mathfrak{f}, \beta, y \rangle \models P$
$\langle \mathfrak{f}, \beta, x \rangle \models [\pi]P$	iff	$\langle \mathfrak{f}, \beta, x \rangle \models \neg \langle \pi \rangle \neg P$
$\langle \mathfrak{f}, \beta, x \rangle \models \langle \pi \cup \sigma \rangle$	iff	$\langle \mathfrak{f}, \beta, x \rangle \models \langle \pi \rangle P$ or $\langle \mathfrak{f}, \beta, x \rangle \models \langle \sigma \rangle P$
$\langle \mathfrak{f}, \beta, x \rangle \models \langle \pi; \sigma \rangle$	iff	$\langle \mathfrak{f}, \beta, x \rangle \models \langle \pi \rangle \langle \sigma \rangle P$
$\langle \mathfrak{f}, \beta, x \rangle \models \langle \pi^* \rangle$	iff	for some $n \in \omega$ $\langle \mathfrak{f}, \beta, x \rangle \models \langle \pi^n \rangle P$
$\langle \mathfrak{f}, \beta, x \rangle \models \langle P? \rangle Q$	iff	$\langle \mathfrak{f}, \beta, x \rangle \models \langle \sigma \rangle P$ and $\langle \mathfrak{f}, \beta, x \rangle \models \langle \sigma \rangle Q$

One may be easily deceived by this calculus in thinking that the addition of $\cup$, * and ? is just a nice notation but does not add any strength since can can always replace these symbols. But this only true if the Kleene-star is not present. With the Kleene-star, this calculus is quite powerful and any of the symbols makes a genuine contribution to it. The program operations have direct analogues in Kleene-algebras. To see this quite clearly we will deliberately confuse modal operators with the abstract relation they stand for. Thus the following abbreviations are justified

$$\begin{array}{lcl} (\Diamond \cup \blacklozenge)p & = & (\Diamond p) \cup (\blacklozenge p) \\ (\Diamond \circ \blacklozenge)p & = & \Diamond(\blacklozenge p) \\ \Diamond^* p & = & (\langle R^* \rangle p) \text{ where } \Diamond \text{ is based on } R \end{array}$$

It is easy to see that if $\langle\pi\rangle$ is the modality corresponding to the program π and the relation R_π in the dynamic frame, then $\langle\pi\rangle\langle\sigma\rangle$ corresponds to $\pi;\sigma$ and this corresponds to the relation $R_\pi \circ R_\sigma$; and $\langle\pi\rangle \cup \langle\sigma\rangle$ corresponds to $\langle\pi \cup \sigma\rangle$ and to $R_\pi \cup R_\sigma$. Finally, π^* corresponds to R_π^*.

The test operator intertwines the boolean valuation and the relation structure that is created over f by the basic relations, because it creates new relations. The relation that corresponds with $P?$ is $\Delta \cap \beta(P) \times \beta(P)$. So, $\langle x, y\rangle \in R_{P?}$ iff $x = y$ and $x \in \beta(P)$. This test therefore does not allow to look at another world; rather it allows us to cut some process in programs if certain conditions are not met at a state.

6.3 Schematic Grammars

In the course of the development of grammatical formalisms it has been realized that simple context free grammars are deficient in two respects. One is that the actual rules do not display the regularities of the language succinctly enough; the other is that natural languages transcend the generating power of context free grammars. As for the latter observation it is hard to find such instances and the arguments are subtle, so we will refrain to develop this point right now. But the first objection in itself is strong enough to call for different tools. Let us look at an example. We know that subject and verb agree in person and number. So, rather than being forced to write several rules elaborating on the basic rule

$$\mathsf{S} \to \mathsf{NP} \quad \mathsf{VP}$$

we would like to write one rule that encapsulates the full consequences of the agreement condition. This can be achieved by introducing *variables* into the rules. In addition we need to import the attribute-value mechanism in order to express exactly over what values the variable is allowed to range. The basic rule is then written as follows.

$$\mathsf{S} \to \mathsf{NP}\begin{bmatrix} \textsc{per}: & \mathsf{X} \\ \textsc{num}: & \mathsf{Y} \end{bmatrix} \quad \mathsf{VP}\begin{bmatrix} \textsc{per}: & \mathsf{X} \\ \textsc{num}: & \mathsf{Y} \end{bmatrix}$$

Such examples are known from definite clause grammars. We call a rule where variables occur *schematic* and a grammar that contains such rules a *schematic grammar*.

It is actually not these types of agreement rules that motivated for us the introduction of schematic rules but the need to capture very complex phenomena (e. g. cumulative head-movement) which are responsible for the non-context freeness of languages. The reason is twofold. First, closer inspection of inflectional systems shows that agreement in feature values accounts only for half of the phenomena involved, the other half being a result of more complex mechanisms which do not admit a formulation as schematic rules. Second, the finite character of inflection means that the agreement system can be unravelled into conventional rules, and so from a purely theoretical standpoint agreement presents the solvable case of grammar evaluation, because we find the results on perfect grammars applicable to this case. Thus, only when the use of variables abbreviates a potentially infinite set of rules is there a need to introduce variables – for our purposes at least.

But how can variables range over infinitely many values? In the example with inflectional agreement we have seen that the variable – as it ranges over atomic values – can realize only a finite number of different choices. Therefore, only when the attribute value formalism involved admits certain infinities in the definition of AVMS can the variables range over infinitely many alternatives, because they range over AVMS. This is important. This means, namely, that morphological and lexical alternations cannot be the source of this infinity; hence it must be nearness restrictions, that is, non-local dependencies and similar phenomena that involve nodes in the tree with no a priori fixed distance. This means also that the proposals of (**?**) of layering the language with which to express syntactical facts on top of the language of AVMS is completely misguided. If we were to do this, it would be a pure coincidence that elements can exert non-trivial influence on each other, because the feature language internal to the nodes influences the syntactic environments of these nodes only by mediation of the rules. Consequently, the information stored in the AVM can only be interpreted in view of the full rule system.

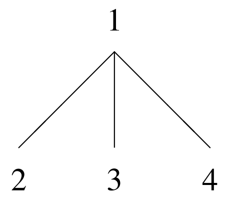
Our view is opposite. Rather than letting the rules decide the meaning of the internal AVMS, we consider their interpretation fixed a priori, similar to the slashes in categorial grammar. [1] So, we will assume that there is a language internal to nodes which allows to code certain external facts about this node. And that this language uses primitives whose interpretation is fixed a priori. The grammatical rules do not necessarily fall out as corollaries of this interpretation, as in categorial grammar, but nevertheless need to conform with the interpretation given to these symbols. So, taking the categorial grammar, A/B expresses the fact that the node combines exclusively with a B (which must be on

[1]It is worthwile observing that the meaning of the slashes in categorial grammar is actually not as clear as it appears to be. The Geach rule, for example, distorts the picture quite drastically. Without it, A/B just codes the fact that the item needs a B to the right to give a mother A. However, if we can also combine A/B with B/C to have a mother A/C – as in Geach rule –, this interpretation needs to be suspended. The analogy with modus ponens in logic, which has driven the development of categorial grammar a long way, has twisted the intuition that the slashes code nearness information and given way to an interpretation of slashes as meta-level statements of syntactic reducibility. The latter might be fine as such, but does not square with the official propaganda of categorial grammar.

the right) to form a constituent of category A. Thus, a rule $A \rightarrow A/B \quad B$ conforms with this interpretation, while $A \rightarrow A/B \quad C$ does not when $C \neq B$. The difference with categorial grammar is that we do not assume the information at the node to be total, that is, we reject the view that the lexical items contain enough information to code the entire syntax. Rather, there are global facts of the language, expressed by rules, which are supplemented by facts which the syntactic items supply locally. In fact, GB assumes that lexical items draw only from a finite source of distinctions, while the syntactic architecture of a language results from various global principles of syntax. It is this view which we are going to formalize with *geometrical grammars*.

6.4 The Orientation Language

With respect to trees there are four fundamental relations which allow to locate elements with respects to each other. These correspond to the notions of being *daughter of*, *mother of*, *the left sister of* and *the right sister of*. We denote them by $\Diamond_{\uparrow}$, $\Diamond_{\downarrow}$, $\Diamond_{\leftarrow}$, $\Diamond_{\rightarrow}$. We illustrate them with a binary tree:



In this tree

$$\begin{array}{lcl}
\Diamond_{\uparrow} & = & \{\langle 2,1\rangle, \langle 3,1\rangle, \langle 4,1\rangle\} \\
\Diamond_{\downarrow} & = & \{\langle 1,2\rangle, \langle 1,3\rangle, \langle 1,4\rangle\} \\
\Diamond_{\leftarrow} & = & \{\langle 3,2\rangle, \langle 4,3\rangle\} \\
\Diamond_{\rightarrow} & = & \{\langle 2,3\rangle, \langle 3,4\rangle\}
\end{array}$$

Notice that $\Diamond_{\uparrow} = \Diamond_{\downarrow}^{\smile}$ and $\Diamond_{\leftarrow} = \Diamond_{\rightarrow}^{\smile}$. So, $\Diamond_{\uparrow}P$ is true at a node iff it has a mother and its mother satisfies P; $\Diamond_{\downarrow}P$ is true at a node iff it has a daughter and one of its daughters satisfies P. $\Diamond_{\leftarrow}P$ true at a node iff there is a sister immediately to the left and it satisfies P; $\Diamond_{\rightarrow}P$ is true a node iff there is a sister immediately to the right and it satisfies P. Various other relations can be defined from these primitives using the usual operations.

RELATION	TRANSLATION
$\Diamond_{\downarrow}^{*}$	dominates
$\Diamond_{\downarrow}^{+}$	properly dominates
$\Diamond_{\uparrow}^{*}$	is dominated by
$\Diamond_{\uparrow}^{+}$	is properly dominated by
$\Diamond_{\leftarrow}^{+} \cup \Diamond_{\rightarrow}^{+}$	sister of
$\Diamond_{\uparrow}^{*} \circ \Diamond_{\rightarrow}^{+} \circ \Diamond_{\downarrow}^{*}$	precedes
$\Diamond_{\uparrow}^{*} \circ \Diamond_{\leftarrow}^{+} \circ \Diamond_{\downarrow}^{*}$	succeeds
$\Diamond_{\uparrow}^{*} \cup \Diamond_{\downarrow}^{*}$	overlaps with
$\Diamond_{\uparrow}^{*} \circ (\Diamond_{\leftarrow}^{+} \cup \Diamond_{\rightarrow}^{+}) \circ \Diamond_{\downarrow}^{*} \cup \Diamond_{\uparrow}^{+} \cup \Diamond_{\downarrow}^{+}$	is different from

We denote by **Ot** the orientation language, which is based solely in the programs $\Diamond_{\uparrow}$, $\Diamond_{\downarrow}$, $\Diamond_{\leftarrow}$ and $\Diamond_{\rightarrow}$ in addition to the features. With some effort it is possible to axiomatize the logical theory of finite trees with these orientation modalities. This logic of finite trees is called **Olt**. We observe that the behaviour of $\Diamond_{\downarrow}^{+}$ can be captured by Gödels logic **G**. This logic is characterized by the axiom

$$\blacklozenge p. \rightarrow .\blacklozenge(p \wedge \neg\blacklozenge p)$$

It can be shown that this axiom implies transitivity, and it says then that for any value for p if p is possible via $\blacklozenge$, there is a successor satisfying p and no successor, direct or indirect, satisyies p again. The behaviour of $\Diamond_{\uparrow}^{+}$, $\Diamond_{\leftarrow}^{+}$, $\Diamond_{\rightarrow}^{+}$ can be captured by **G**. In addition, $\Diamond_{\uparrow}$, $\Diamond_{\leftarrow}$, $\Diamond_{\rightarrow}$ satisfy $\mathbf{Alt_1}$. Furthermore, there are axioms to epress the fact that $\Diamond_{\leftarrow}$ and $\Diamond_{\rightarrow}$ as well as $\Diamond_{\uparrow}$ and $\Diamond_{\downarrow}$ are converse modalities, namely

$$p \rightarrow \neg\Diamond_{\leftarrow}\neg\Diamond_{\rightarrow}p, \quad p \rightarrow \neg\Diamond_{\rightarrow}\neg\Diamond_{\leftarrow}p$$

It remains to fix axiomatically the relation between $\Diamond_{\leftarrow}$ and $\Diamond_{\uparrow}$. Here we use the axioms

$$\Diamond_{\uparrow}p \wedge \Diamond_{\leftarrow}q. \rightarrow .\Diamond_{\leftarrow}(q \wedge \Diamond_{\uparrow}p)$$

$$\Diamond_{\uparrow}p \wedge \Diamond_{\rightarrow}q. \rightarrow .\Diamond_{\rightarrow}(q \wedge \Diamond_{\uparrow}p)$$

By standard techniques it can be shown that these axiom are equivalent on refined frames to the elementary conditions that if $x\Diamond_{\uparrow}y$ and $x\Diamond_{\leftarrow}z$, then $z\Diamond_{\uparrow}y$ and if $x\Diamond_{\uparrow}y$ and $x\Diamond_{\rightarrow}z$, then $z\Diamond_{\uparrow}y$. (Again notice that we use the modality to denote the corresponding relation.)

Theorem 6.4.1 *A finite frame is a* **Olt***-frame iff it is a tree.*

Proof. Clearly, $\Diamond_{\leftarrow}$ has to satisfy $\mathbf{Alt_1}$, and if it does, the frame is based on a relation where $\Diamond_{\leftarrow}$ offers for each point at most one alternative. Similarly, by correspondence the postulates $p. \rightarrow .\neg\Diamond_{\rightarrow}\neg\Diamond_{\leftarrow}p$ and $p. \rightarrow .\neg\Diamond_{\leftarrow}\neg\Diamond_{\rightarrow}p$ force the relation $\Diamond_{\rightarrow}$ to be the converse of $\Diamond_{\leftarrow}$. The postulates

$$\Diamond_{\uparrow}p \wedge \Diamond_{\leftarrow}q. \rightarrow .\Diamond_{\leftarrow}(q \wedge \Diamond_{\uparrow}p)$$

$$\overset{|}{\Diamond} p \wedge \Diamond\!\!\!\!-\, q. \rightarrow .\Diamond\!\!\!\!-\,(q \wedge \overset{|}{\Diamond} p)$$

induce the property that sisters to a daughter node of x are daughter nodes as well. Thus, these daughters are linearly ordered by $-\!\!\!\!\Diamond^{+}$ as well as $\Diamond\!\!\!\!-^{+}$. $\dashv$

This theorem does not extend to infinite frames; this is the source of great complications. If, namely, we could enforce models to be finite the logic would be decidable. But this is not so, and the infinite models can be rather nonstandard. We will return to this problem in a short while.

Let us now turn to a slightly different class of frames. These frames arise naturally when we allow reentrancy in trees. This can be desirable in dealing with movement. The idea would be that both the mother of the moved constituent and the trace left by it dominate the same structure. The verb, requiring an object of a certain type to its right, can check the satisfaction of this requirement simply by inspection of the trace rather than indirectly by inspection of the moved phrase. The consequence of this is that the structures admit several mother nodes per group of sisters. Such structures will be called **daglod**s (directed acyclic graphs with linearly ordered daughters). Their logic, called **Olg**, is **Olt** minus the $\mathbf{Alt_1}$ postulate for $\overset{|}{\Diamond}$. These structures are based on two relations, $\overset{|}{\Diamond}$ and $\Diamond\!\!\!\!-$ such that $x \overset{|}{\Diamond} y \Diamond\!\!\!\!-\, z$ and $x \overset{|}{\Diamond} y$ with $z \Diamond\!\!\!\!-\, y$ imply $x \overset{|}{\Diamond} z$. So, mothers must share the same group of sisters.

6.5 Geometric Grammars versus Schematic Grammars

If rules can be expressed in the orientation language, it is not hard to imagine that one can actually eliminate rules altogether in favour of the local conditions on trees they induce. This is actually in line with the philosophy of the GB research. The idea is that rather than using rules one should search for principles which allow to derive these rules in their conjunction. In addition, there should be certain finite parameters that need be set to define a particular core grammar. This latter feature of GB shall be ignored here. However, the principles are an important feature of GB – and in fact all other grammar formalisms. Despite claims to the contrary these formalisms use rules as well as principles, so there is never a formulation that uses rules or principles exclusively. This has to do with a certain conceptual economy. Notice namely that rules in a grammar are taken disjunctively, it makes no sense to speak of the local condition a rule places on the local trees, rather than the condition the totality of rules imposes on the local trees. Likewise, a principle does not translate into a rule, because it is a condition on the local trees admitted by the grammar, and thus affects all rules. Principles act conjunctively, rules disjunctively. There is now great emphasis on the advantage conjunctive principles have for grammars and parsing and it sometimes sounds as if a disjunction can be rejected on principled grounds. [2]

[2]Just consider the case of (Rizzi, 1990). He argues quite strongly for a non-disjunctive formulation of the ECP but the definition of minimality itself is disjunctive. The whole discussion, like that of (Chomsky,

This is not justifiable, since the disjunction represented by the disjunction are there, and whether they can be avoided by a suitable revision of the overall theory depends of the case at hand. A basic core of alternatives in the form of real disjunctions does in fact always remains – in the form of $\overline{X}$-syntax, for example.

To summarize, rules act disjunctively and principles act conjunctively. In actual syntactical frameworks there is a natural balance between rules and conditions on representations. Nevertheless, there is theoretical possibility of a strictly rule based grammar and a strictly principle based grammar. The latter is exemplified by *geometric grammars* while the former is exemplified by *schematic grammars*. A schematic grammar allows the use of variables in rules; however, unlike usual definite clause grammars these variables have quite special interpretation, namely they express *geometrical structure sharing*. Similarly, a geometric grammar is a finite set of sentences from the orientation language. Likewise, these sentence may contain variables, and these variables express structure sharing. It needs to be said quite clearly what we mean by structure sharing. There are basically two interpretations. The first is that structure means *constituent*. So, a variable under that interpretation stands for a constituent, and using the same variable means talking about the same constituent. The second interpretation is that structure means the whole accessible structure, so not only the downwardly accessible structure (as in the first interpretations), but also the structure accessed by going up (and back down again). The latter interpretation is easier to handle logically, because here a variable acts as a standard logical variable. Let us illustrate this with an example. Suppose we state via a formula in the orientation language that a trace has an antecedent which is subjacent. Using the orientation language we can express this as follows

$$\mathsf{TR} \cap \mathsf{X}. \to .\langle sub \rangle \mathsf{X}$$

Here, *sub* is the relation of subjacency, and X a variable. We assume here that traces have the structure of the antecedent (thus are not feature- and structureless), the only difference being that they are phonetically empty. The latter is signalled already by the feature TR. This postulate, if expressed this way will force the antecedent and the trace to be generators of the same open constituent; namely, the generating nodes are different, but everything else is shared. This is a consequence of the interpretation of X. We can namely substitute *any* formula for it. So, if the trace is dominated by a $\overline{\mathsf{V}}$, as for an object trace, so must be the antecedent, because we can substitute $\Diamond(\mathsf{V} \cap \mathsf{BAR}{:}1$ for X. And similarly, if the trace has a verb as a left sister, so has the antecedent, by substituting $\Diamond(\mathsf{V} \cap \mathsf{BAR}{:}0)$. And so on.

The consequence is that the structures compatible with this interpretation are not really ordered trees but daglods. This consequence is emphasized in HPSG, where actual identity of the structure which is shared is considered an explanation for the fact that

1986a) is an exercise in propaganda shadow boxing. Unification by underlying principles is sometimes not distinguishable from a unification by disjunction of cases. It is a question of the terminological primitives, and in the effect boils down to the same.

information about this structure flows both ways between the nodes accessing this structure. However, there is an apparent overload in the information that we transfer under this interpretation, for it is only the structure of the trace that needs to be accessed by the antecedent, not the environment this trace lives in. The axiom above does therefore not really (under our intentions) speak of structure as of the orientation language, but constituent structure. Constituent structure can be associated with special expressions of the orientation language.

Definition 6.5.1 *A* ***constituent*** *in the orientation language is an expression of the form* $A \cap \Diamond\!\!\!\shortmid\;M$*, where* A *is nonmodal (= free of modalilties) and* M *free of* $\overset{|}{\Diamond}$*. A* ***constituent variable*** *is a variable which admits as subsitution instances only constituents.*

Thus, constituent variables are more appropriate linguistically, while ordinary variables are more suitable from a logical point of view. This difference is accentuated in the following definitions.

Definition 6.5.2 *A* ***geometrical grammar*** *is a quadruple* $\langle F, \Sigma, \Omega, \Lambda\rangle$*, where* F *is a finite set of features,* $\Sigma, \Omega \in \mathfrak{Fr}(F)$*, and* Λ *a finitely axiomatized extension of* ***Olg****.*

Definition 6.5.3 *A* ***schematic grammar*** *is a quadruple* $\langle F, \Sigma, \Omega, R\rangle$ *where* F *is a finite set of features,* $\Sigma, \Omega \in \mathfrak{Fr}(F)$*, and* R *a finite set of rules* $M \to N_1 \dots N_m$*, where* M, N_i *are* ***Ol****-formulae over* F *possibly containing constituent variables.*

These rather long winded definitions have a common core consisting of a set F of basic features forming the boolean algebra $\mathfrak{Fr}(F)$ from which both the start symbol Σ and the terminal symbol Ω are drawn. The difference is that the structures are defined in a geometric grammar by axioms concerning the structure of the daglods (not ordered trees) while a schematic grammar is like an ordinary definite clause grammar rule with the possibility to use complex symbols (written with all four operators $\overset{|}{\Diamond}$, $\Diamond\!\!\!\shortmid\;$, ${}^{-}\!\Diamond$, $\Diamond\!{}^{-}$) while the variables themselves are constituent variables. The latter can be considered an optical trick. This way we ensure that the structures the grammar generates conform to the interpretation we must assign to the symbols. Namely, the grammar generates trees, so no reentrancy is possible. On the other hand shared variables mean shared structure. The puzzle is resolved if structure means constituent structure.

6.6 From Rules to Geometric Axioms and Back

First we deal with the simple case of a boolean grammar. A (memory) F-grammar grammar $\mathbb{S}$ can be seen as a machine that produces models for the orientation language. Not only does it generate a structure (the trees) but it also produces a labelling at each node in case we have full access to it's internal states, that is, in case it is rational. We can characterize the models that are generated by $\mathbb{S}$ by a single axiom. The axiomatic extension of **Olt** by this axiom will characterize all and only the finite models of **Olt** that $\mathbb{S}$ generates. Consider for the sake of concreteness that $\mathbb{S}$ contains only one rule, namely $\mathsf{a} \rightarrow \mathsf{b}_1 \ \mathsf{b}_2$. Then consider the following formula

$$\mathsf{a} \wedge \Diamond_{\downarrow}(\neg\Diamond_{\leftarrow}\top \wedge \mathsf{b}_1 \wedge \Diamond_{\rightarrow}(\mathsf{b}_2 \wedge \neg\Diamond_{\rightarrow}\top))$$

This says that a is mother and one step below a is a b_1, and it is leftmost among the nodes covered by a. Furthermore, it has a sister to the right satisfying b_2, which itself has no right sister. Exactly as the local condition expressed by the rule. Now consider a translation of rules $\rho \mapsto \rho^t$. If $\rho = \mathsf{a} \rightarrow \mathsf{b}_1 \ldots \mathsf{b}_n$ then

$$\rho^t = \mathsf{a}. \wedge .\Diamond_{\downarrow}(\neg\Diamond_{\leftarrow}\top \wedge \mathsf{b}_1 \wedge \Diamond_{\rightarrow}(\mathsf{b}_2 \wedge \Diamond_{\rightarrow}(\mathsf{b}_3 \ldots \wedge \Diamond_{\rightarrow}(\mathsf{b}_n \wedge \neg\Diamond_{\rightarrow}\top)\ldots)))$$

Now put

$$\mathbf{Ax}(\mathbb{S}) = \bigvee_{\rho\in R} \rho^t. \wedge .\neg\Sigma \rightarrow \Diamond_{\uparrow}\top. \wedge .\neg\Omega \rightarrow \Diamond_{\downarrow}\top$$

Proposition 6.6.1 (First Characterization) *Let $\mathbb{S}$ be a rational F-grammar. A finite, fully labelled tree with labels from the labelling of $\mathbb{S}$ satisfies* **Olt**.**Ax**($\mathbb{S}$) *iff it is generated by $\mathbb{S}$.*

This theorem is the easy part of a description of context-free grammars. It shows that a grammar is effectively describable by a single **Ol** axiom. We will later show that to a certain extent this theorem has a converse; any **Ol** axiom can be *coded* into context-free rules in such a way that the grammar consisting with these rules conforms with this axiom. However, it is still open whether the First Characterization extends to non-rational grammars. The difficulty is that we have to use only the symbols from F and so cannot 'spy' into $\mathbb{S}$.

In a similar way we can reduce schematic rules from a schematic grammar. To translate a schematic rule $\rho = \mathsf{A} \rightarrow \mathsf{B}_1 \ldots \mathsf{B}_m$ we use the same translation, but now A, B_i are geometric formulae from the orientation language. The variables originally act as constituent variables, but are now read as structure variables, and so the translation returns a geometric grammar. The problem is now to compare the structures that the two grammars describe. But it is intuitively clear that a daglod can be disentangled in such a way that it yields a uniquely determined ordered tree. Namely, if two nodes x_1 and x_2 share the same open constituent C we split C into two isomorphic copies C_1 and C_2 and insert C_1 below x_1 and C_2 below x_2. This procedure is repeated until all nodes have unique mothers. We then say that a geometric grammar is **finitely equivalent** to a schematic grammar if the daglods it admits are the same as the trees generated by the schematic grammar – after disentangling. Notice, however, that the geometric also admits infinite models, which the schematic grammar does not. Thus a stricter notion of equivalence is necessary if true equivalence is wanted. We say that the geometric grammar is **strongly equivalent** if it is determined by its finite frames. This means that it admits infinite frames, but the logical theorms of that grammars can be evaluated by just looking at the finite models.

Theorem 6.6.2 (Rules to Geometry) *Let* $\langle F, \Sigma, \Omega, R\rangle$ *be a schematic grammar. Then* $\langle F, \Sigma, \Omega, \mathbf{Olg}(R^t)\rangle$ *is a weakly equivalent geometric grammar.* $\dashv$

Theorem 6.6.3 (Geometry to Rules) *Let* $\langle F, \Sigma, \Omega, \mathbf{Olg}(\Phi)\rangle$ *be a geometric grammar containing an axiom* $\neg\Diamond^k\top$ *for some* k*. Then there exists a weakly equivalent schematic grammar.*

Proof. The proof is by reference to the Coding Theorem. This theorem says that it can be guaranteed for any formula ϕ of the orientation language that it is distributed by means of schematic rules exactly according to its semantic interpretation. So we can write a schematic grammar that distributes the formulas Φ according to their semantic interpretation. Now take a rule $\mathsf{A} \to \mathsf{B}_1 \dots \mathsf{B}_m$, $m \leq k$, of this grammar and replace it by

$$\mathsf{A} \cap \bigcap \Phi \to \mathsf{B}_1 \cap \bigcap \Phi \dots \mathsf{B}_m \cap \bigcap \Phi$$

The totality of these rules form the rule set R of the schematic grammar. $\dashv$

It is illustrative to see the exact mechanism of this proof. The axiom $\neg\Diamond^k\top$ says that each nodes has at most k daughters. Without such an axiom we would not have a grammar of finite type. Now we take the feature system $\mathfrak{Fr}(F)$ and write down all possible rules $\mathsf{F} \to \mathsf{G}_1 \dots \mathsf{G}_m$, $m \leq k$, with $\mathsf{F}, \mathsf{G}_i \in \mathfrak{Fr}(F)$. Successively, we code the subformulas of Φ into these rules. That this is possible (and how it is) is shown by the Coding Theorem in 7.7.7. This means that the rules still have no content, that is, no restrictions are worked in. We only know that when a formula appears, e. g. $\Diamond^+(\mathsf{F}_2 \to \Diamond\Diamond\mathsf{F}_1)$, that it appears exactly at those nodes where it should according to its interpretation. This ensures that if we now add $\bigcap \Phi$ at all nodes we can be sure that the postulates of Φ are obeyed.

6.7 The Formal Evaluation of Grammars

The conversion of grammars into logics allows to restate certain important problems concerning grammars as logical problems, and there is some hope that the tools from modal logic can supply sufficiently satisfying answers to them. The first problem is that of generative capacity. If $\mathbb{G}$ and $\mathbb{H}$ are two schematic grammars, can we decide whether they are generatively equivalent, or whether one is stronger than the other? We understand by generative power not the generative power in terms of strings (whch is known to undecidable) but rather generative capacity in terms of structures, ordered trees that is. Thus, by replacing these grammars by their geometric counterparts we can rephrase this problem as a condition on models for the underlying logic of the geometric grammars. Let namely $\langle F, \Sigma, \Omega, \Lambda \rangle$ be a geometric grammar. The set F is implicit in the language of Λ, so is of no real significance. The start symbol codes the axiom $\neg\Diamond\top \to \Sigma$, and the terminal symbol codes the axiom $\neg\diamondsuit\top \to \Omega$. Thus, by adding these two postulates to Λ we obtain a slightly stronger logic which contains all the conditions on models of the geometric grammar. [3] Thus a geometric grammar is an extension of **Olg** with finitely many constants by finitely many axioms. Given two geometric grammars, Λ, Θ, over the same language, the generative power of the corresponding schematic grammar is decided by the set of finite models these logics admit. Since they also admit infinite models, it is necessary to postulate here that the logics are determined by their finite models. Suppose that they are; then we can decide both $\Lambda \subseteq \Theta$ and $\Lambda = \Theta$, and consequently the corresponding problem about the generative power of the schematic grammars. Namely, the theorems of Λ are recursively axiomatized and so enurable. Furthermore, for a finite frame it is decidable whether it is a model for Λ. We cane therefroe enumerate the frames for Λ, and with them the models, and so the non-theorems of Λ. Now take an axiom of Θ, θ. We can decide $\Lambda \vdash \theta$, and so we can decide $\Lambda \supseteq \Theta$. Reversing the roles we can decide whether $\Theta \supseteq \Lambda$, and so we can also decide whether $\Lambda = \Theta$. All this is subject to the condition that the logics are strongly equivalent to their schematic grammars, that is, that they have the finite model property.

Thus the finite model property is the key property here. One might suggest to weaken this to decidability, which would still leave the evaluation problem with an answer, but in that case we cannot really speak of a good representation of the schematic grammar by the corresponding logic. It should be relatively easy to come up with logics which fail to have the finite model property and so are not stronlgy equivalent to their schematic grammars. It is most likely not even decidable which logics have fmp. Nevertheless, since human languages use only a fraction of the possibilities it might very well be that this fraction is a bundle of logics which have the finite model property. This problem, however, needs to be left open. An important subcase is the base logic **Olg**.

Problem 6.7.1 *Does* ***Olg*** *have the finite model property?*

A positive answer to this question would immediately show that all context free grammars are strongly represented by their geometric grammars. Namely, context free grammars use a finite symbol resource and can thus be axiomatized without the use of variables, that is, with a constant formula. So a context free grammar reduces to an extension of **Olg** by a

[3]The presence of start and terminal symbol is technically required in the definition of a geometric grammar only because we need to make sure we can identify them. From a pruely technical point of view it is possible to do without them.

constant axiom A. Now, from **Olg**(A) we can deduce a formula ϕ exactly if we can deduce

$$(\Diamond \cup \Diamond \cup \Diamond \cup \Diamond)^{*}A. \rightarrow .\phi$$

from **Olg**. (See (Kracht, 1993) and (Kracht, 1995) for similar arguments.) Therefore, if **Olg** has the finite model property, so does **Olg**(A).

There is thus a marked difference between our approach to syntax and the one proposed in (**?**). Stabler uses full predicate logic to formalize GB theories and throws away all chances of obtaining decidability results outright. Here, the language is a fragment of dynamic logic and the question of decidability is not a priori negatively answered. But even if it turns out that the general evaluation problem is undecidable for **Olg**, there are good arguments to prefer the orientation language over predicate logic. One is the local character of the primitives $\Diamond$, $\Diamond$, $\Diamond$, $\Diamond$ and the full correspondence with rule based mechanisms and axiom or constraint based mechanisms based on the orientation language. Such a close correspondence cannot be found with predicate logic because the latter is too amorphous. The fact that just anything of interest can be coded makes it unlikely that useful results will appear. With our approach, however, the question of representing facts in grammars is still nontrivial – because the mechanisms are restrictive – but we might be rewarded with clearer insight into the grammars themselves and – hopefully – with substantial decidability results.

Chapter 7

Spreads and Codes

7.1 Definition and Examples

Suppose that we have a set F of features (mostly finite) and an F-tree $\mathfrak{T}$. We can distinguish in $\mathfrak{T}$ two types of sets; an **internal set** is one that can be defined by a proposition over $\mathbb{F}$, that is, $S \subseteq T$ is internal if there exists a boolean expression A in terms of the features from $\mathbb{F}$ such that $S = [[\mathsf{A}]]$. If S is not internal then it is **external**. Moreover, consider an abstract set $\mathfrak{S}$, that is, a function $\mathfrak{S}$ that returns for each F-tree $\mathfrak{T}$ a set $\mathfrak{S}(\mathfrak{T}) \subseteq \mathfrak{T}$. $\mathfrak{S}$ is **internal** if there exists a boolean F-term A such that for all F-trees $\mathfrak{T}$ $\mathfrak{S}(\mathfrak{T}) = [[\mathsf{A}]]_{\mathfrak{T}}$. Examples are the abstract set $F_1 \cap F_2$ which selects for each labelled tree $\mathfrak{T}$ the set of nodes which are both in F_1 and F_2. (We will not distinguish thoroughly between a term $\textsc{a}$ and the abstract set $[[\mathsf{a}]]$.) Simply speaking we require to have a formula that selects in each labelled tree exactly the set that we have singled out via $\mathfrak{S}$. If $\mathfrak{S}$ is not internal, it is called **external**. This distinction between internal and external is quite rough; we can for instance think of other ways to identify sets uniformly in trees, for example by use of predicate logic. The set of branching is clearly dedinable by a first-order formula. So, being external can be split into being first-order definable and not being first-order definable. The interest in such sets and abstract sets lies in the relation with grammars. We can view grammar writing as a process whereby external sets become internalized. This means that we start with a slot grammar (on no symbols!) and gradually add the labels. BUt we add them in such a way that their distribution is exactly as we want it. This is, however, quite difficult to achieve and therefore formal tools that can be of help are highly desirable. The way we proceed here is by specifying external sets in some suitable way and then internalize them using an effective algorithm. It is namely often comparatively easy to say in informal terms what particular distribution a label has while it is hard to write a grammar that achieves this.

We will provide mechanisms that translate such specifications into rules. Moreover, we will calibrate the strength of the conditions that can be imposed on such sets if this translation is at all possible. This is not a trivial thing to do. We can for example select the set of all nodes whose depth is a prime number. Since this latter (abstract) set never defines a regular path set on sets of trees for which there is no upper bound on the depth, there is no chance whatsoever of writing a cfg that has an internal abstract set of this type. This raises the question of the exact limits of internalization. This limit can be determined quite precisely. We will show that the abstract sets that are definable via the orientation language **Ol** with labels from F are exactly those sets that can be internalized in this sense. This is the content of the Coding Theorem.

Before we can enter the proof of the Coding Theorem, we will broaden the notion of an abstract set. First, we will allow for different choices of sets to be instantiated at one and the same tree; moreover, we will allow to specify a sequence of sets rather than a single set. The latter will prove to be an irrelevant generalization, but is at least from an intuitive standpoint helpful.

Definition 7.1.1 *Let $\langle \mathbb{T}, \ell \rangle$ be a labelled tree; an* ***n-extension*** *of $\langle \mathbb{T}, \ell \rangle$ is a sequence $\langle \mathbb{T}, \ell, S_1, \dots, S_n \rangle$ where $S_i \subseteq T$, $i \leq n$. An* ***n-spread*** *is a collection of n-extensions over (finite) trees. Finitely, an* ***n-spread over*** *$\mathfrak{T}$, a set of labelled trees, is a collection of n-extensions of the trees from $\mathfrak{T}$.*

Given an n-extension $\mathcal{E} = \langle \mathbb{T}, \ell, S_1, \dots, S_n \rangle$ of a labelled tree $\mathcal{T}$ we can form an equivalent labelled tree $\mathcal{E}^\sharp$; equivalent in the sense that it contains the same information as $\mathcal{E}$. Namely, take a set $G = \{\mathsf{G}_i | i \leq n\}$ of labels such that $F \cap G = \emptyset$ and define assignments

$$\mu_i(x) = \begin{cases} \mathsf{G}_i & \text{if} \quad i \in S_i \\ -\mathsf{G}_i & \text{if} \quad i \notin S_i \end{cases}$$

Put $\ell^\sharp = \ell \otimes \mu_1 \otimes \dots \otimes \mu_n \in \mathfrak{L} \otimes \mathfrak{Fr}(n)$. Then $\mathcal{E}^\sharp = \langle \mathbb{T}, \ell^\sharp \rangle$. $\mathcal{E}^\sharp$ is a $F \cup G$-tree. Similarly any n-spread over a set $\mathfrak{T}$ of trees is reduced to a set of labelled trees; only we need to use the same labels for the same S_i throughout. The extensions are thus made homogeneous. If a distinction is needed between $\mathcal{E}$ and $\mathcal{E}^\sharp$ we will refer to the former as a **set-extension** and to the latter as a **label-extension**.

The formal definition of spreads leaves a lot of room for arbitrariness; the question that naturally arises is how to extract appropriate spreads from linguistic data. Based on some formal considerations we will formulate a principle at the end of this chapter that allows to construct optimal spreads.

7.2 Codability of Spreads

We assume that $\mathfrak{S}$ is an n-spread over the set of finite F-trees for a finite set F of labels; that $\langle\mathbb{G}, \epsilon\rangle$ with $\mathbb{G} = \langle\Sigma, \mathfrak{L}, R\rangle$ is a silent boolean F-grammar. $\mathfrak{S}$ defines a spread on $\mathrm{Yd}(\mathbb{G})$ which in turn gives rise to a set $\mathrm{Yd}(\mathbb{G})^\sharp$ of trees with labels in $\mathfrak{L} \otimes \mathfrak{Fr}(n)$.

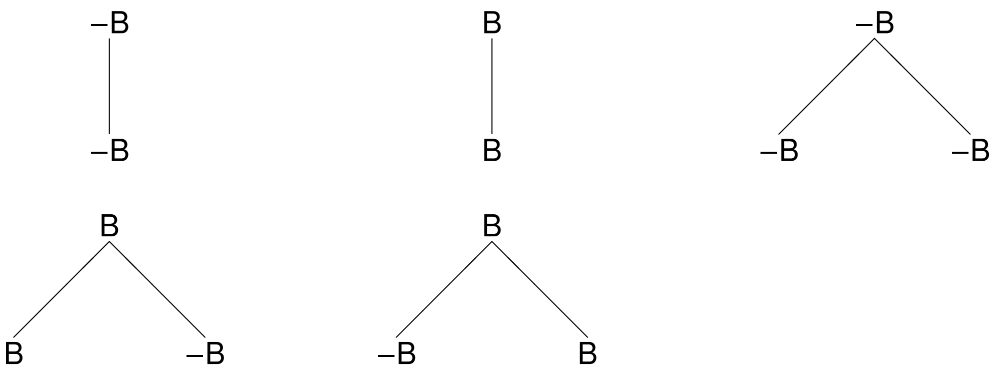
Definition 7.2.1 *Suppose that $\mathcal{G} = \langle\mathbb{G}, \epsilon\rangle$ is a silent boolean F-grammar and* $\mathrm{Yd}(\mathbb{G})$ *the set of finite trees generated by $\mathbb{G}$. Let $\mathfrak{S}$ be an n-spread. We say that $\mathfrak{S}$ is* ***memory codable*** *in $\mathbb{G}$ if there is a $F \cup G$-grammar $\mathcal{H} = \langle\mathbb{H}, \zeta\rangle$ with finite memory with $G = \{\mathsf{G}_1, \dots, \mathsf{G}_n\}$ such that $\mathfrak{E} = \langle\mathbb{T}, \ell, S_1, \dots, S_n\rangle \in \mathfrak{S}$ exactly if $\mathcal{G} \otimes \mathcal{H} \gg \langle\mathbb{T}, \ell \otimes \mu\rangle$ and $S_i = [[\mathsf{G}_i]]$ for all i. The silent grammar $\mathcal{H}$ is called the* **code** *of $\mathfrak{S}$ in $\mathbb{G}$. We call a spread $\mathfrak{S}$* ***memory codable*** *simpliciter if it is memory codable in every grammar.*

This calls for some exegesis. The basic idea is that coding a spread into the grammar $\mathcal{G}$ means an amalgamating it with another grammar in such a way that the spread is internalized. The spread tells us that over the set of F-trees certain (unknown) features have to be distributed in exactly the way presented. Thus we need to expand our trees by some more features in order to make room for the new sets in the spreads. It is then our task to distribute them according to the spread. Thus we identify each S_i with a feature in the extended language, using also the new labels if necessary. Thus we have to first upgrade $\mathcal{G}$ into a $F \cup G$-grammar, but with no conditions associated with the G-features. This means passing on from the F-trees generated by $\mathcal{G}$ to the set of all $F \cup G$-trees whose F-reduct is generated by $\mathcal{G}$. Next we tensor $\mathcal{G}$ with another silent grammar $\mathcal{H}$. The whole point of this is that $\mathcal{G}$ and $\mathcal{H}$ share the burden of distributing labels. $\mathcal{G}$ is responsible for the F-features, $\mathcal{H}$ is responsible for the G-features. Seen this way the construction is symmetric in $\mathcal{G}$ and $\mathcal{H}$. But there is an intuitive difference that we want to capture, namely that spreads refine an already existing grammar $\mathcal{G}$. This is reflected in the fact that we require $\mathcal{H}$ to be of finite memory, thus, in fact, context free.

Example 1. Let $\mathfrak{Brn}$ be the *branch spread*. This is the spread of all 1-extensions where S_1 is simply any branch of the tree. (To be precise here: $\mathfrak{Brn}$ contains for all trees all sets that are branches. So $\mathfrak{Brn}$ is not necessarily an abstract set since there is generally more than one branch in a tree.) This spread is indeed codable. To see this, let us try to define how the rules of the grammar have to be enriched with the symbol B such that it will be distributed along a single branch. Take a rule $\rho = \mathsf{L} \to \mathsf{M}_1 \dots \mathsf{M}_n$. We must define a set of extensions $\rho^\sharp$ of ρ that schedule B. $\rho^\sharp$ naturally splits into two sets of extensions. The first is when the mother node has B and the second is when the mother node has $-\mathsf{B}$. Both cases must be dealt with. B means being in the spread; thus, if the mother node bears $-\mathsf{B}$ it is clear that neither of the daughters are B. Consequently, there is only one extension of ρ of this type, namely, $\mathsf{L} \otimes -\mathsf{B} \to \mathsf{M}_i \otimes -\mathsf{B} \dots \mathsf{M}_n \otimes -\mathsf{B}$. However, if the mother node in a rule is B then exactly one of the daughter nodes must be B as well. In this case we must add a new rule for each daughter, $\mathsf{L}\otimes\mathsf{B} \to \mathsf{M}_1\otimes\mathsf{B} \quad \mathsf{M}_2\otimes-\mathsf{B} \dots \mathsf{M}_n\otimes-\mathsf{B}$,

$\mathsf{L}\otimes\mathsf{B} \to \mathsf{M}_1\otimes -\mathsf{B} \quad \mathsf{M}_2\otimes\mathsf{B}\ldots\mathsf{M}_n\otimes -\mathsf{B}$ etc. The start symbol of the grammar has to $\Sigma\otimes\mathsf{B}$, because cannot afford not to select the root as part of the branches!

If $\mathbb{G}$ is at most two branching (= of expansivity at most 2), then to code the branch spread it suffices to tensor $\mathbb{G}$ with the following grammar $\mathbb{RRN}$. For other types it is quite clear how the code looks like.



Example 2. Given a n-spread $\mathfrak{S}_1$ and an m-spread $\mathfrak{S}_2$ we can form an $m+n$-spread $\mathfrak{S}_1\otimes\mathfrak{S}_2$ which consists of all $m+n$-extensions $\langle\mathfrak{T},\overline{S}_1,\overline{S}_2\rangle$ such that $\langle\mathfrak{T},\overline{S}_1\rangle\in\mathfrak{S}$ and $\langle\mathfrak{T},\overline{S}_2\rangle\in\mathfrak{T}$. Suppose that $\langle\mathbb{H}_1,\zeta_1\rangle$ codes $\mathfrak{S}_1$ into $\mathbb{G}$ and $\langle\mathbb{H}_2,\zeta_2\rangle$ codes $\mathfrak{S}_2$ into $\mathbb{G}$. Then $\langle\mathbb{G}_1\otimes\mathbb{G}_2,\zeta_1\cup\zeta_2\rangle$ codes $\mathfrak{S}_1\otimes\mathfrak{S}_2$ into $\mathbb{G}$.

Example 3. Furthermore, we can form from the n-spread $\mathfrak{S}$ an n-1-spread by dropping one of the sets, for example, S_n. So, define

$$\mathfrak{S}^{\natural(n)} = \{\langle\mathfrak{T},S_1,\ldots,S_{n-1}\rangle | (\exists S_n\subseteq T)(\langle\mathfrak{T},S_1,\ldots,S_n\rangle\in\mathfrak{S})\}.$$

Suppose $\mathfrak{S}$ is coded in $\mathbb{G}$ by $\langle\mathbb{H},\zeta\rangle$. Let $\zeta^{\natural(n)} = \zeta \restriction (F\cup G) - \{\mathsf{G}_n\}\rangle$. Then $\mathfrak{S}^{\natural(n)}$ is coded by $\langle\mathbb{H},\zeta^{\natural(n)}\rangle$.

We say that $^{\natural(i)}$ is a **forget** operation. We say that $\mathfrak{T}$ is an **expansion** of $\mathfrak{S}$ and $\mathfrak{S}$ is a **contraction** of $\mathfrak{T}$ if there is a series of forget operations $^{\natural(i)}$ which transform $\mathfrak{T}$ into $\mathfrak{S}$. This forget operation brings us to the actual coding problem that we sketched at the beginning.

A code, being a silent grammar, need not be rational, that is, we might need more unobservables to guarantee the successful distribution of the spread-features in addition to the unobservables that we need to distribute to F-features of the original grammar. We say, then, that a spread is **memory free codable** if the coding grammar is memory free, or rational. So memory free codability means that there are no additional features needed to encode the spread. However, by adding some finite set of extra features we can turn a silent grammar into a rational grammar, simply by making the unobservables observable. That this set is finite is a consequence of the requirement that the code must have finite memory. Let the unobservables be $\mathsf{X}_1, \dots, \mathsf{X}_k$. Consider then an expansion of $\mathfrak{S}$ into a spread $\mathfrak{T}$ that distributes the X_i for $i \leq k$ over $\mathbb{G}$-trees exactly as the grammar rational expansion of $\mathcal{H}$ does; then by definition $\mathfrak{T}$ is memory free codable and $\mathfrak{S}$ is a contraction of $\mathfrak{T}$.

Proposition 7.2.2 *$\mathfrak{S}$ is memory codable iff there exists an expansion of $\mathfrak{S}$ which is context free codable.* $\dashv$

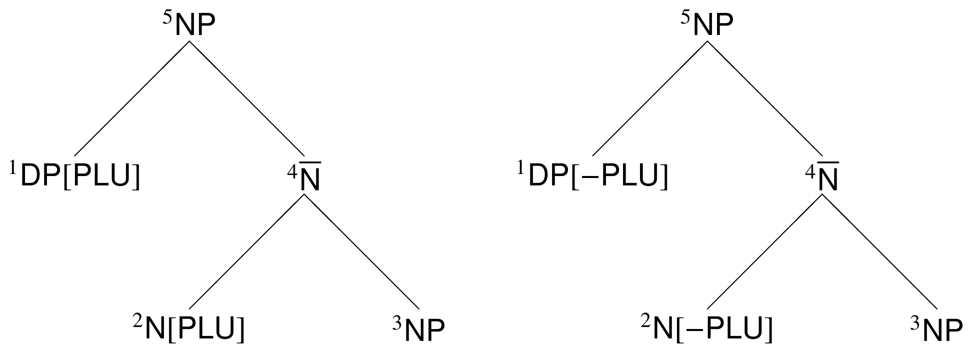
7.3 Covariance and Codability

This section will introduce some basic terminology and prove certain essential theorems on codability. Even though their correctness is intuitively quite clear, the necessary consequences on the construction of grammars are quite far reaching. We start by investigating the current practice of defining spreads in linguistics. Spreads are underdetermined by the data so that there is some room for arbitrary decisions. This leap from data to theory is normally taken without justification and linguists are sometimes prone to present their particular version of feature distribution as necessitated by the facts, so that the data would support their theory rather than the theory that is being criticized. We will demonstrate the complexity of this issue by a seemingly innocuous example, namely *agreement*. Before I can start presenting the kernel principle of agreement feature sharing in GB let me say passim that it is rather disappointing how many textbooks on GB with pompous titles – usually employing phrases like *the theory of grammar* – fail to mention agreement or treat it in the way it deserves. A notable exception is (**?**). One might get the impression that this was just a minor issue of syntax. This is clearly not so, and GB has incorporated principles that are supposed to reflect agreement. The problem is that they are not recognized by the reader as such, nor is their role explained in the textbooks. If one digs hard enough one finds, for example, this.

SPECIFIER-HEAD-AGREEMENT (SHA) The head of a phrase agrees with its specifier.

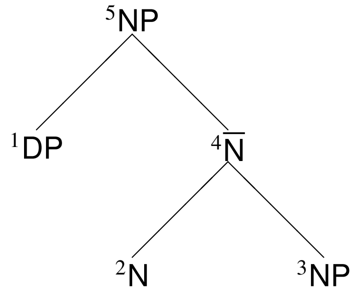
Notice that this principle as stated is quite empty; it does not clarify what it means to agree and is typically stated without explanation in the books that I consulted. There is in fact no fixed set of features uniform over all basic categories that can be called *agreement features*. For example, when specifier and head of a complementizer agree then it is quite plausible that they agree in different features than if the specifier and head of a noun phrase agree. This is simply a reflex of different classification schemes for the different heads. But let us not get distracted by this issue. We agree on the following convention. To ‘agree’ means to carry the same gender, case and number features, these are also commonly called *agreement features*. The SHA is responsible for the distribution or spreading of such features in a syntactic tree. It is certainly not the only principle since it does not account for the fact that adjectives agree with the nouns they modify, but grammar books of the kind that I have been looking at do not treat this case at all. We will not question, therefore, the correctness of the SHA, rather we want to use it as a guinea-pig for studying codability. To simplify things at the beginning we do as if the only agreement feature was PLU which distinguishes plural from non-plural (= singular in English, but not necessarily in all languages, e. g. Old Greek and Sanskrit).

To see the effect of SHA we unravel an NP.



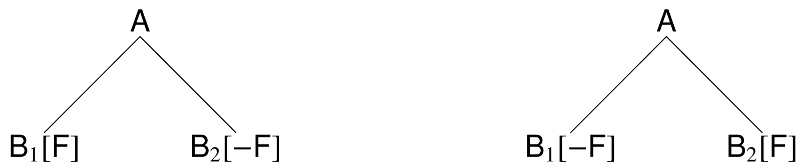
We are following strict $\overline{X}$-syntax; nothing is implied by writing the specifier as DP rather than DET. Neither do we question whether NPs should better be analysed as DPs. As usual, non-assigned features are unspecified and may therefore be assigned at random. Thus both of the above trees represents 8 different choices of full assignments. To distinguish between the base grammar and the spread we have used the notation NP[PLU]. This stands for the fact that in this given extension the node is in the set S_1. When we have successfully coded the spread we will change notation to NP $\cap$ PLU.

Let us begin by noting that the formulation of the SHA presupposes $\overline{X}$-syntax of some form and so the question of codability can be asked meaningfully only if we assume the grammars to extend $\overline{X}$-syntax. But let us forget that problem as well and assume a boolean grammar generating the base trees such a the ones above and let us try and code SHA into it. To this end we fix a tree generated by the base grammar $\mathbb{G}$ such as



This will be our *base tree*. If $\mathcal{E}$ is an extension of $\mathcal{T}$ we let the **index** $ind_{\mathcal{E}}(x)$ of a node x in $\mathcal{T}$ be the set of numbers $i \leq n$ such that $x \in S_i$. Alternatively, we let the index be a sequence of $+$'s and $-$'s indicating whether or not $x \in S_i$. In the first extension $ind_{\mathcal{E}}(1) = \{1\}$, or, $ind_{\mathcal{E}}(1) = \langle + \rangle$; in the second extension we have $ind_{\mathcal{E}}(1) = \emptyset$, or, in the alternative notation $ind_{\mathcal{E}}(1) = \langle - \rangle$. The **choices of** x in $\mathcal{T}$ are all those indices for which there exists an extension in the spread giving the node this index: $ch_{\mathcal{T}}(x) = \{ind_{\mathcal{E}}(x) | \mathcal{E} \in \mathfrak{S}, \mathcal{E} \text{ extends } \mathcal{T}\}$. In the base tree all nodes have the choices $\{\langle + \rangle, \langle - \rangle\}$. Yet there is an intuitive difference between the nodes 1 and 2 and the others. The difference is that the choices cannot be independently be instantiated at the node 1 and the node 2. To formulate this we define the **covariance** of x and y to be the set $cov_{\mathcal{T}}(x; y) = \{\langle ind_{\mathcal{E}}(x), ind_{\mathcal{E}}(y) \rangle | \mathcal{E} \in \mathfrak{S}, \mathcal{E} \text{ extends } \mathcal{T}\}$. x and y are called **independent** if $cov_{\mathcal{T}}(x; y) = ch_{\mathcal{T}}(x) \times ch_{\mathcal{T}}(y)$; otherwise they are called **dependent**. In case of dependency a choice made at x may not be compatible with any other choices made at y. So, 1 and 2 are dependent since $cov_{\mathcal{T}}(1; 2) = \{\langle +, + \rangle, \langle -, - \rangle\}$ while $ch_{\mathcal{T}}(1) \times ch_{\mathcal{T}}(2) = \{\langle +, + \rangle, \langle +, - \rangle, \langle -, + \rangle, \langle -, - \rangle\}$. All other pairs of nodes are independent.

If $\mathfrak{S}$ is codable in $\mathbb{G}$ we can give for each rule ρ of $\mathbb{G}$ a set of extensions ρ^+ such that the so extended grammar $\mathbb{G}^+$ generates the trees of $\mathfrak{S}$. Thus it should be intuitively clear that the source of dependency must lie in the rules, that is, the local trees. A rule ρ seen as a local tree $\mathsf{A} \to \mathsf{B}_1 \ldots \mathsf{B}_n$ has a fixed set of extensions. To give an example (taking here the binary case) let there be only



In this case B_1 and B_2 are *dependent* and this dependency results from the fact that the other combinations of choices are not legitimate rules. The dependency of B_1 and B_2 may extend to a dependency between the daughters of these nodes, but it need not, e. g. in case that the daughters are independent from their mothers.

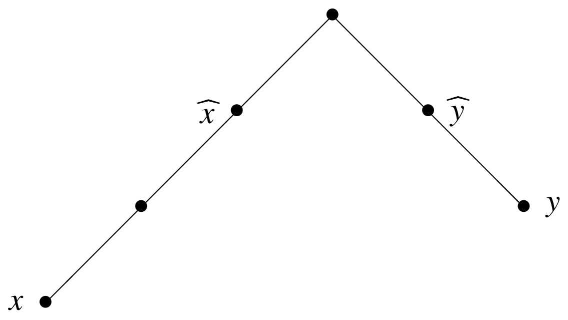
On the other hand, it is clear that if a node y dominates a subtree $\downarrow y$ and $\mathcal{E}$ and $\mathcal{F}$ are extensions of $\mathcal{T}$ such that $ind_{\mathcal{E}}(y) = ind_{\mathcal{F}}(y)$ then we may replace $(\downarrow y)_{\mathcal{E}}$ in $\mathcal{E}$ by $(\downarrow y)_{\mathcal{F}}$. This observation is used to prove the next lemma.

Lemma 7.3.1 (Independency) *Suppose that $\mathfrak{S}$ is codable in $\mathbb{G}$ and that $\mathbb{G} \gg \mathcal{T}$. Let x, y, z be nodes of $\mathcal{T}$ such that $x \nleq y$ and $z \leq y$. If x and y are independent, so are x and z.*

Proof. Let $\mathfrak{i} \in ch_{\mathcal{T}}(x)$ and $\mathfrak{j} \in ch_{\mathcal{T}}(z)$. We have to show that $\langle \mathfrak{i}, \mathfrak{j} \rangle \in cov_{\mathcal{T}}(x; z)$. This shows that $ch_{\mathcal{T}}(x) \times ch_{\mathcal{T}}(z) \subseteq cov_{\mathcal{T}}(x; z)$; the other inclusion, $cov_{\mathcal{T}}(x; z) \subseteq ch_{\mathcal{T}}(x) \times ch_{\mathcal{T}}(z)$, is trivial. Take an extension $\mathcal{F}$ such that $ind_{\mathcal{F}}(z) = \mathfrak{j}$ and let $\mathfrak{k} := ind_{\mathcal{F}}(y)$. By independency of y from x there exists an extension $\mathcal{E}$ such that $ind_{\mathcal{E}}(y) = \mathfrak{k}$ and $ind_{\mathcal{E}}(x) = \mathfrak{i}$. Now substitute $(\downarrow y)_{\mathcal{F}}$ for $(\downarrow y)_{\mathcal{E}}$ in $\mathcal{E}$. This defines an extension $\mathcal{G}$; $\mathcal{G} \in \mathfrak{S}$ because $\mathfrak{S}$ is codable. Moreover, $ind_{\mathcal{G}}(x) = \mathfrak{i}$ since $x \nleq y$ and $ind_{\mathcal{G}}(z) = ind_{\mathcal{F}}(z) = \mathfrak{j}$. Thus $\langle \mathfrak{i}, \mathfrak{j} \rangle \in cov_{\mathcal{G}}(x; z)$. ⊣

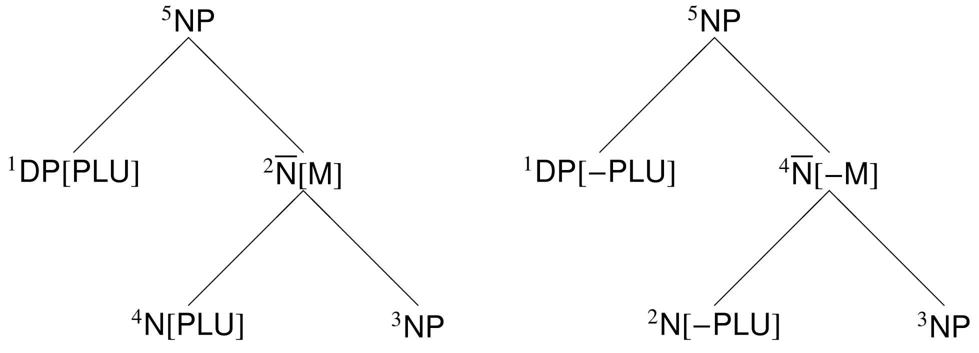
Now call x and y **close** if they are within the same local tree; alternatively, x and y are close if they are $\perp$-mates. The Independency Lemma shows how to trace dependency back to dependency between close nodes. Consider two nodes x, y. Suppose $x \nleq y$. If y does not $\perp$-command x then there exists a $\widehat{y} > y$ such that $\widehat{y}$ does not dominate but $\perp$-command x and by the previous lemma x and $\widehat{y}$ are dependent if x and y are. Similarly we find that if $y \nleq x$ and y does not $\perp$-command x then for the lowest $\perp$-commanding ancestor $\widehat{x}$ of x, $\widehat{x}$ depends on y. Let us call $\widehat{x}$ and $\widehat{y}$ the **close ancestors** of x and y if $\widehat{x}$ is the least ancestor of x $\perp$-commanding y and $\widehat{y}$ the least ancestor of y $\perp$-commanding x. Then as a result of our investigations we find the following.

Lemma 7.3.2 (Dependency) *$\mathfrak{S}$ is codable in $\mathbb{G}$ only if for every base tree $\mathcal{T}$ and every pair of nodes x and y can only be dependent if their close ancestors $\widehat{x}$ and $\widehat{y}$ are dependent as well.* ⊣

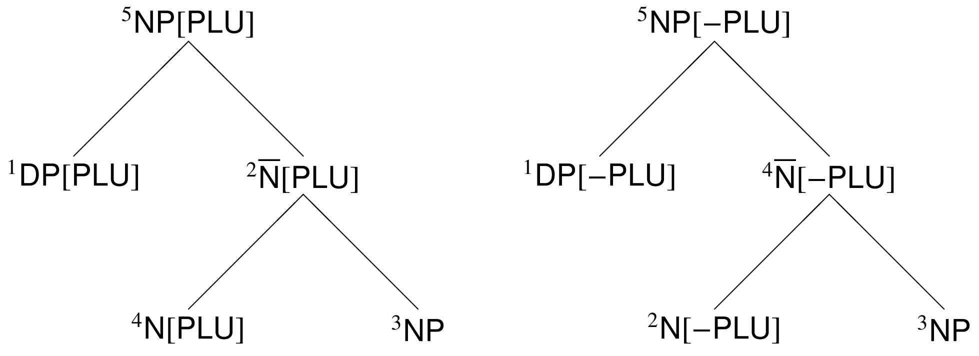


This theorem deserves attention even though it is not hard to prove. Let us note, namely, that we do not need that for a pair of dependent nodes there is a common ancestor on which both depend. So, if we introduce the *dependency domains* of a spread in a labelled tree to be subsets such that for each pair of points there is a chain of dependencies running from one to the other, then these domains are convex sets with the top being removed; thus, they are either ragged cones or open ragged cones.

By this theorem we see that SHA is not codable in its present form. Namely, 1 and 2 are dependent but their close ancestors, 1 and 4, are not. Thus we need to have a memory. It is in this case enough to have a memory of size 2 which can be realized by adding a single feature M. The trees must be specified for M as follows.



At this point we may pause to investigate the question whether it was a good choice to ignore the value of PLU at the other nodes. Notice namely that (1) for all interior nodes the value of morphophonological features can only be hypothesized while for the terminal nodes it can be attested by direct grammatical data and (2) if one node is independent from all other nodes and we have full choice of instantiating the features we can with no harm stipulate some particular instantiations as the only legitimate ones. Thereby we restrict the spread artificially but in this case the result is markedly superior. We stipulate now that the agreement feature is spread over the whole phrase with the exception of the complement. (Whether it will be spread over adjuncts will not be discussed now; we are at the moment only considering our example base tree.)



We can say then that with the interpretation of PLU in mind we have used the feature PLU as a *memory for itself*. This almost exaplains itself; theoretically, as in GPSG we need to distinguish between marking the value of the agreement features and marking the fact that in the local structure the agreement head has such and such agreement features; the latter is abbreviated by AGR: PLU and AGR: – PLU. Although distinct in interpretation, it is a question of economy in the grammar (as well as the relation between grammatical data and the necessitation of a particular analysis) whether the two can be identified. In this particular case this can be done. This spread is codable (modulo other unseen feature instantiations for adjuncts etc.) and we can immediately read off a boolean grammar coding it.

However, for the sake of concreteness let's look at a more complex issue, namely *disagreement* constructions (see (Baker, 1992)). (I claimed in an earlier version that disagreement is not memory free codable, but have actually discovered that it is. This shows how tricky this subject is when it comes to such issues.) In Mohawk, discussed in the quoted article, both the sentence [A] and [B] require dual marking for the verb, while in many European languages only in the [B] sentence the verb is non-singular (i. e. plural, since they have no dual, mostly).

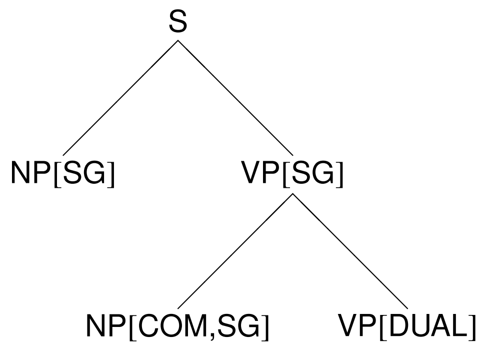
[A] I danced with Mary.

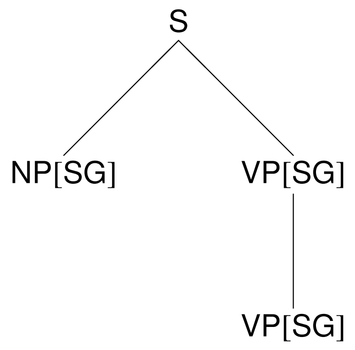
[B] Mary and I danced.

If we would simply let PLU (or DUAL) percolate from the subject down to the verb we would have the situation as in European languages; the verb decides on the basis of the plural feature of the subject. If the situation is as in Mohawk, the verb must receive two pieces of information: the marking of the subject and the marking of the comitative phrase. How can we achieve this? If we have to posit a phrase structure analysis of the following kind

$$\mathsf{S} \to \mathsf{NP}\ \ \mathsf{VP}\ \ \mathsf{NP[COM]}$$

then it is evident that the two pieces can be passed to the VP without memory because the information is close (in the technical sense). If we have a binary branching analysis with the comitative phrase being the VP-adjunct, we do the following





Other assignments are done similarly. The idea is that the information of the agreement status of the subject is passed on until the adjunct is reached; it's agreement features are then merged with the ones of the subject to yield the VP-agreement features. One might have objections from an intuitively point of view, but formally this analysis is as good as any other.

7.4 Join Spreads and Reusability

The discussion of agreement leads us to another important question in codability namely that of *reusability*. In inflectional languages it is not necessary to mark the verb for agreement at all; we can rely on case to tell us which NP serves as which argument of the verb. So agreement is a kind of luxury, or, to use a neutral term, introduces redundancy into the system. If there is indeed no agreement or only one type of agreement, either subject-verb or object-verb then the plural marking on the verb does not need to discriminate these types of agreement and the feature PLU is enough. If, however, both types of agreement are present the verb needs to realize them differently; we consequently have to split PLU into PLU-S and PLU-O; we might be tempted to understand them as abbreviations of [PLU, CASE : NOM] and [PLU, CASE : ACC] but if these two features meet at the verb they clash because their conjunction is contradictory. On the other hand, PLU itself does not seem to need a distinction at the NP itself whether it is PLU-S or PLU-O. The NP needs to be marked only for plural simpliciter. Only outside of the NP it matters whether it is a PLU-S or a PLU-O.

All this applies only to a single clause, however. Outside the clause these features can be used again with different meaning. So, while PLU-S has one and only one abstract meaning, namely to specify morphological agreement of the verb with its subject, its concrete use in different clauses provides different contexts and different concrete meanings even within the same tree. It is a general feature of grammatical mechanisms that outside of their nearness domains they can be used again for same purpose but with different instantiations. A quite surprising example will be discussed now, that of *binding*. We know that binding is mediated via $\perp$-command. Technically, x binds y if x is coindexed with y and $\perp$-commands it. To code this we need a memory unless y $\perp$-commands x by chance as well. But for the moment we will concentrate not on this question but on the question of reusability. Let us therefore ignore the coindexing clause, in fact, let us indiscriminately spread the index of x over its $\perp$-domain. Then being coindexed with x is the same as being $\perp$-commanded with x. Already this poses problems. If, namely, the only thing we have access to is the indexing, it is impossible to recover x. If x and y are $\perp$-mates then they have the same $\perp$-domains (they are $\perp$-co-heads in our terminology). So we need to distinguish the origin of the index in some way. This detail is frequently overlooked in GB. Here, for example, is a version of the *i*-within-*i*-filter in a recent textbook ((Haegeman, 1991)):

The *i*-within-*i*-filter $^*[_{A_i} \ldots B_i \ldots]$

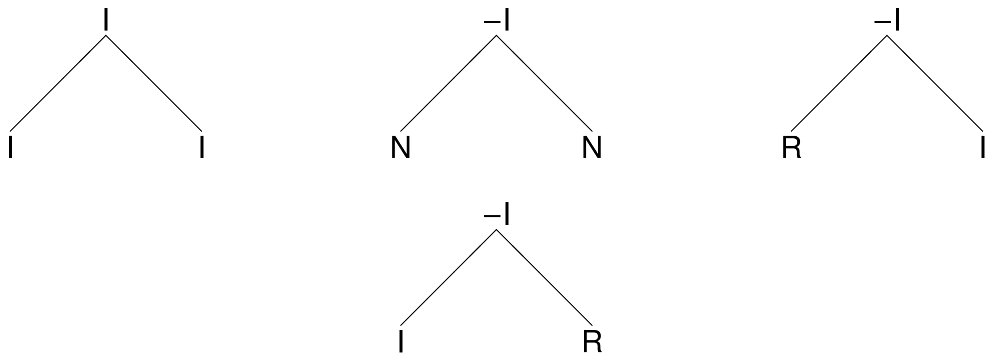
What is *not* meant here is that B may not be coindexed with A in the case where A $\perp$-commands B or that they may not bear the same index. What is forbidden by this is that A and B create the same index. There is no way to make this precise without making the distinction between index creation and coindexation. We must therefore assume that an index IND is accompanied by a feature REF selecting items at which the index is instantiated, or better, which are co-referential with the given index. The *i*-within-*i*-filter will be encapsulated in the following postulate

Unique Indexation $^*\mathsf{REF} \cap \mathsf{IND}$

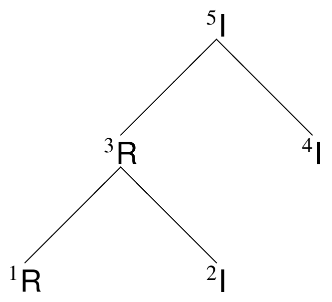
This leaves us with three other possibilities:

$$\begin{array}{lcl}
\mathsf{N} & : & -\mathsf{IND}\cap-\mathsf{REF} \\
\mathsf{I} & : & \mathsf{IND}\cap-\mathsf{REF} \\
\mathsf{R} & : & -\mathsf{IND}\cap\ \mathsf{REF}
\end{array}$$

Thus we have the following rules:



Specifically, these rules pass I on to all daughters and no more referents will be created under the same name. Hence, when I is initiated it co-occurs with a sister R and the I is henceforth distributed exactly over the $\perp$-command domain of the referent. However, since the referent itself does not carry the index, the very same index can be used inside of it, so that the following structure is quite possible.

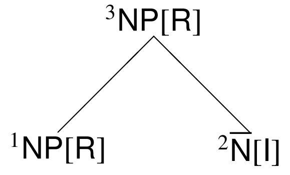


However, the same index means different things. At 2 it means being coindeced with 1 and at 4 it means being co-indexed with 3. This is not as funny as it first appears. In predicate logic one and the same variable can be used in different contexts to mean different things, likewise in computer languages like Pascal and Algol. This is illustrated by the formula below.

$$(\forall x)(\phi(x) \wedge (\exists x)(\psi(x) \rightarrow \phi(x)))$$

Consider now the following phrase

[Peter's$_i$ best friend]$_i$

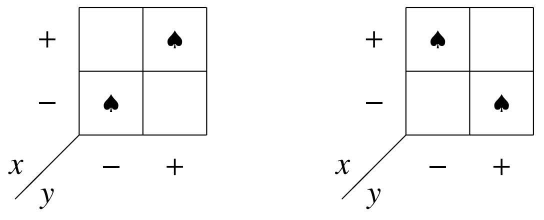


While the referent attribute REF may be instantiated at both 1 and 2 the two need not mean the same thing. However, *best friend$_i$* bearing the index i is 'coindexed' with *Peter*, so that the object-variable in *friend*(x_1, x_2) has to be the same as for *Peter*. The first variable, however, being the referent of the higher NP, may or may not be co-referential with x_2. The two way identity: identity because of being the same variable and identity because of coreference – are the two ways in which a referent REF and an index IND can be identical. The first case arises wheen the REF-bearing node $\perp$-commands the IND-bearing node and the second if it does not.

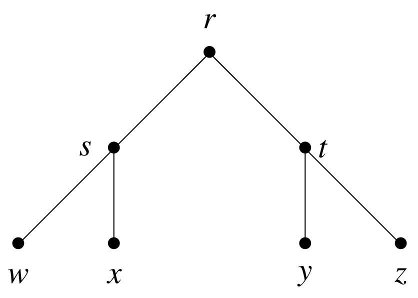
7.5 Principles of Spread Construction

With the terminology and results developed so far we can return to the problem of extracting spreads from the linguistic data. By data we understand the some abstractly defined map $^\flat$ that relates grammatical trees with observable structures. The best observable is the physical string, either as spoken or as written down on paper. This raises interesting questions. Namely, the assignment of the agreement features to nodes in a tree is supported from the data only inasfar as it can be documented in morphological correlations between words; and the latter are determine only the properties of the leaves of the syntactic tree. Even though we consider intuitively the property of being *plural* to be attributable not only to simple nouns but also to noun phrases, this is not a priori guaranteed to be a necessary consequence of the morphological facts. In fact, the deeper one engages in this question the less clear it becomes. Nevertheless, we can formulate principles that are quite viable and give a postiori explanations for our intuitions.

We remain with agreement, especially number agreement. Let us assume, that number agreement can be isolated as an independent phenomenon, i. e. that number agreement is uncorrelated with gender agreement and let us try to define an optimal spread for number agreement. How are we to proceed? First let us be clear in the terminology. Given a single feature F we say that two nodes x and y in a labelled tree are **positively F-correlated** or F-**agree** if $cov_{\mathfrak{T}}(x; y) = \{\langle -, -\rangle, \langle +, +\rangle\}$, and that they are **negatively F-correlated** if $cov_{\mathfrak{T}}(x; y) = \{\langle -, +\rangle, \langle +, -\rangle\}$.



For a set F of features we say that x and y F-**agree** if they F-agree for all $\mathsf{F} \in F$. Agreement means agreeing with respect to a certain set F. Agreement is a special type of correlation and indeed the most simple one to be handled. Agreement with respect to a set of features can always be decomposed as agreement with respect to a single feature. Agreement is also a transitive relation. If x and y F-agree and y and z F-agree, then x and z F-agree as well. Call $A \subseteq T$ an **agreement set** if it is the set of all nodes agreeing with a given node x. Let A be an agreement set and z the smallest node such that $\downarrow z \supseteq A$. Call D an **agreement domain** if D is of the form $\downarrow z \cap \uparrow A - \{z\}$. Alternatively, D is an agreement domain if it is the dependency domain of A. Even though agreement sets are disjoint, agreement domains need not be.



In the tree shown above assume that w and y F-agree, x and z F-agree, but that w and x are F-uncorrelated. It follows that we have the agreement sets $\{w, y\}$ and $\{x, z\}$ and the agreement domains $\{w, s, t, y\}$ and $\{x, s, t, z\}$. I have to stress here that we do not have a fully specified spread, whether or not the interior nodes of the tree agree or are uncorrelated cannot be answered. Rather, we want to develop means to reconstruct (or construct) the spread at the interior nodes through this analysis.

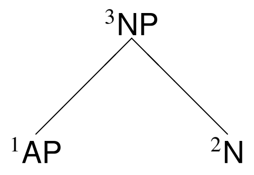
It is clear that overlapping dependencies mean that the feature F is not straightforwardly reusable. The problem is created at the interior points at which the domains overlap. These points have a problem in storing the feature value for two different agreement sets. It is clear that this cannot be done without memory. But assume that we have an A Priori here, namely that on all trees of the grammar the number of agreement sets containing a given point is bounded by α. Then we can actually disentangle these domains by introducing new feature algebras. If previously we had the algebra $\mathfrak{F}$, we now have $\mathfrak{F} \otimes \mathfrak{F} \otimes \ldots \otimes \mathfrak{F}$, α times. The situation is quite similar to that of the index algebra, and I can recommend reading the section on interpreting the tensor product once again at this point. If $\alpha = 2$, the feature F is split into $\mathsf{F} \otimes 1$ and $1 \otimes \mathsf{F}$, which are identical morphologically, but are now hypothesized as different. A possible case in point are AGR-S and AGR-O.

After disentangling the agreement domains we postulate the following spread. With respect to a tree $\mathfrak{T}$ the extension $\langle \mathfrak{T}, S \rangle$ belongs to the agreement spread for F iff for each agreement domain D either $S \cap D = \emptyset$ or $S \supseteq D$. So, a spread set may not split an agreement domain. This is plausible; otherwise the agreement link between certain agreeing leaves is cut somewhere so that they are forced to disagree in the label-extension corresponding to $\langle \mathfrak{T}, S \rangle$. It is now not guaranteed but highly probable that this spread is memory free codable.

Let us demonstrate this procedure with practical examples. We know that adjectives and nouns PLU-agree in Latin. Spefically, we have

$\surd$	mare	nostrum	*	mare	nostra
	$-$	$-$		$-$	$+$
*	maria	nostrum	$\surd$	maria	nostra
	$+$	$-$		$+$	$+$

This leads to the postulating this spread for the parse tree.



1	2	3
−	−	−
−		+
+	+	+
+	+	+

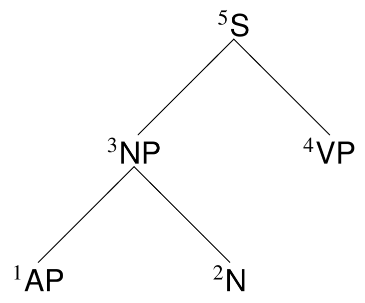
Put into prose this means that although the sisters must agree whether they are positively or negatively PLU, we cannot on the basis of this data conclude that the NP mother agrees with them. This is counterintuitive, but we will see that the conclusion is correct on the basis of the given data. A similar situation is namely agreement between subject and verb.

√	philosophus	ambulat	*	philophus	ambulant
	−	−		−	+
*	philophi	ambulat	√	philosophi	ambulant
	+	−		+	+

The conclucion is the same, namely that subject and verb agree, but the sentence itself does *not* need to agree. This time, however, the conclusion is supported by our intuition. The difference is easily explained by considering more complex phrases.

√	philosophus	illustris	ambulat	*	philophus	illustris	ambulant
	−	−	−		−	−	+
*	philosophus	illustres	ambulat	*	philophus	illustres	ambulant
	−	+	−		−	+	+
*	philosophi	illustris	ambulat	*	philophi	illustris	ambulant
	+	−	−		+	−	+
*	philophi	illustres	ambulat	√	philosophi	illustres	ambulant
	+	+	−		+	+	+

Not only do we see the old agreement between adjective and noun, but also between adjective and verb, noun and verb. This time the principle of spread construction dictates that we assume the noun phrase to agree with the noun, the adjective and the verb; for it is contained in the agreement set of noun and verb. (With $A = \{1, 2, 4\}$ we have $D = \{1, 2, 3, 4\}$.)



This leads to the conclusion that noun phrases are marked for PLU in the same way as their noun head. A similar embedding test shows quite different results. Subject sentences are always singular, be their own subject singular or not. This is quite universally true in all languages. This leaves two options. We can say that sentences are singular throughout, or that it is not meaningful to speak of singular or plural sentences. In the boolean language this cannot be expressed as such. However, there is an analogue of this. Call a point x F-**isolated** if it is independent of all other points in a tree. Call an element in a mereology of a boolean grammar F-**isolated** if all F-points are F-isolated. Then for F-isolated points it makes no sense to speak of their marking for either being positively F or being negatively F. This applies to lexical categories (such as adverbs and other immutable words) and also to non-lexical categories such as sentences. It turns out that on this interpretation sentences are not PLU-isolated, and that we must classify them as *singularia tanta*, i. e. categories which are necessarily singular.

Not everything concerning plural can be analysed in terms of agreement. Consider coordination constructions such as *John and I*, *Either the car or the bicycle*. Analysing the English data, a conjunction *NP and NP* is plural throughout, independently of whether the individual conjuncts are singular or plural. Since plural is connected with a subject consisting of a single individual, this is not surprising and might lead to the conclusion that in coordination constructions the driving factor in the correlation of PLU-marking is the semantics. But this not necessarily so. Consider

I had a constant fight with myself.
Me and myself always disagree.

We do not attempt at settling this question here but let us pursue the syntactic complications for PLU-marking in coordination. Firstly, it is evident that there is no agreement between conjuncts, not even between mother and daughters. In the English case, in conjunctive coordination the mother is independent from it's conjuncts because it is plural. In languages with dual there is a clear case for a *correlation* between mother and daughters but not agreement. We leave it for the reader to verify the details. Notice also that although there is a correlation along each axis compound-left conjunct and compound-right conjunct, the correlation is a function from the marking pair left conjunct-right conjunct to the marking of the compound. This function is as follows.

$$\begin{array}{c|c|c|c|}
\hline
p & p & p & p\\
\hline
d & p & p & p\\
\hline
s & d & p & p\\
\hline
\multicolumn{1}{c}{l \diagup r} & \multicolumn{1}{c}{s} & \multicolumn{1}{c}{d} & \multicolumn{1}{c}{p}
\end{array}$$

This raises the question whether this function is actually determined by the individual correlational data. However, it is compatible for the mother to be plural while the left daughter is singular and the right daughter is not singular. Or, the left daughter is singular and the right conjunct is not singular. With only this correlational data we would conclude that the compound can be plural if both conjuncts are singular – which we must reject. This number system shows that the correlation analysis is good for spotting dependencies, but too weak to represent them faithfully.

7.6 Specification of Spreads

As we noted earlier, a spread is a construction and is as such either externally given or produced by a certain description. For example, the branch spread is the 1-spread where S_1 is a branch of the tree in question. The *mother-of-A* spread is the 1-spread where S_1 collects all mothers of nodes of category A. And so on. Apart from these informal ways to introduce spreads, there are more formal ones. The latter are realized by writing down this specification in a formal language; most suitable in our case is the language **Ol**. For example, the *mother-of-A* is spread is rendered formally as follows. We reserve for S_1 a new feature X_1; the distribution for X_1 is then laid down by the equation $\mathsf{X}_1 = \Diamond\mathsf{A}$. Equivalently, we could have written the axiom $\mathsf{X}_1 \leftrightarrow \Diamond\mathsf{A}$. The latter corresponds more closely to the idea of filtering. Any new feature, here X_1, is in principle free to distribute itself wherever it wants. Thus without the axiom we would have the trivial 1-spread over the trees, where S_1 can be any set. But if we add this axiom, we filter out all extensions of trees in which S_1 does not collect the mothers of A-nodes. So, with basic features $\mathsf{F}_1, \dots, \mathsf{F}_m$ given, we can specify spreads with **Ol**-formulas by writing axioms of the form

$$\begin{array}{lcl}
\mathsf{X}_1 & \leftrightarrow & \phi_1(\mathsf{F}_1, \dots, \mathsf{F}_m)\\
\mathsf{X}_2 & \leftrightarrow & \phi_2(\mathsf{F}_1, \dots, \mathsf{F}_m)\\
\dots & & \dots\\
\mathsf{X}_n & \leftrightarrow & \phi_n(\mathsf{F}_1, \dots, \mathsf{F}_m)
\end{array}$$

But as it stands, this schema is quite restrictive. Consider, for example, the branch spread. There is no single formula over the given features $\mathsf{F}_1, \dots, \mathsf{F}_m$ that allows to define all and only the branches. This is easy to see; namely, since the features are already given, that is, their distribution is fixed, the distribution of the X_i is also fixed. Thus, spreads defined in this way do not allow multiple extensions of the same tree. Hence this schema characterizes abstract sequences of sets rather than spreads. Thus we have to find something else. The solution lies in lifting the restriction that the distributions of the X_i are independent of each other; e. g. we can imagine quite well defining X_2 in terms of X_1, X_3 in terms of both X_1 and X_2; however, this still is deterministic on the same sense. What is creating indeterminacy is when the dependencies displayed by the formulas are circular. We can write for example

$$\mathsf{X}_1. \leftrightarrow . - \Diamond^{+}\mathsf{X}_1 \cup -\Diamond^{+}\mathsf{X}_1$$

This means that X_1 is true at a node iff it is true at none of its sisters; evidently this allows for more than one solution. Indeed, it is checked that this formula defines the branch spread. So we relax the definitional schema in the following way.

Definition 7.6.1 *A n-spread $\mathfrak{S}$ is **Ol-specifiable** iff formulas $\phi_1, \ldots, \phi_n$ exist such that the distribution of X_i, corresponding to S_i, as defined by $\mathfrak{S}$ coincides with the satisfaction of the following equations*

$$\begin{array}{lcl} X_1 & \leftrightarrow & \phi_1(F_1, \ldots, F_m, X_1, \ldots X_n) \\ X_2 & \leftrightarrow & \phi_2(F_1, \ldots, F_m, X_1, \ldots X_n) \\ \ldots & & \ldots \\ X_n & \leftrightarrow & \phi_n(F_1, \ldots, F_m, X_1, \ldots X_n) \end{array}$$

For example, $\mathsf{X}_1 \leftrightarrow \Diamond\Diamond\top)$ specifies the spread *is a branching node*; $\mathsf{X}_1 \leftrightarrow (\Diamond^{*} \circ (\Diamond \circ (-\mathsf{A}?))^{*} \circ (\Diamond \cup \langle\top?\rangle)\mathsf{F}$ specifies the spread *being c-commanded by an F-node*. This needs some checking. We will later show how the usual nearness conditions can be rendered in **Ol**.

The most elegant way to code a spread into $\mathbb{G}$ would be to write a grammar $\mathbb{C}$ that codes $\mathfrak{S}$ simultaneously over all grammars using the same grammar $\mathbb{C}$. But this can evidently not be done; just think of coding the branch spread into grammars whose type is not known. Then we are at a loss as to how large $\mathbb{C}$ has to be chosen because if the expansiveness of $\mathbb{G}$ exceeds that of $\mathbb{C}$ we accidentally 'kill' rules instead of just coding a branch into them. On the other hand we have the feeling that the expansiveness of $\mathbb{G}$ is the only thing we have to monitor, in fact the rules we should put into $\mathbb{C}$ do not individually depend on the type of $\mathbb{G}$, rather we need to know whether or not to put the rules of type m into $\mathbb{C}$. So, while for each type T there is a grammar $\mathbb{C}^T$ directly coding the branch spread into grammars of type T, it is really so that if $T \subseteq U$ then R^T is the set of rules of type T contained in R^U. In order not get bogged down too much into formal details it seems profitable to use a single grammar $\mathbb{C}^{\omega}$ of type $\omega = \{1, 2, 3, \ldots\}$ to code the spread for all grammars. Surely, $\mathbb{C}^{\omega}$ is not finite, but it contains only finitely many rules for any finite subtype grammar. So when $\mathbb{G}$ is of finite type T then $\mathbb{G} \otimes \mathbb{C}$ is certainly of finite type $T' \subseteq T$. In this way we can ignore the problems connected with types. On the other hand, when writing concrete codes we will restrict ourselves to grammars of expansiveness ≤ 2 and specify codes for them only. The general grammar $\mathbb{C}^{\omega}$ can mostly be guessed immediately.

Definition 7.6.2 *A spread $\mathfrak{S}$ is **directly (memory) codable** if there is a grammar $\mathbb{C}^{\omega}$ of type ω such that for any grammar $\mathbb{G}$ of type T $\mathbb{C}^{\omega}_T$ codes $\mathfrak{S}$ into $\mathbb{G}$, where $\mathbb{C}^{\omega}_T$ is the restriction of $\mathbb{C}^{\omega}$ to the subtype T.*

Direct codability can be characterized quite effectively. Notice namely that codes have the property to be downwardly stable.

Proposition 7.6.3 *Let $\mathfrak{S}$ a spread over $\mathbb{F}$-trees, and let $\mathbb{G}$ and $\mathbb{H}$ be $\mathbb{F}$-grammars such that* $\mathrm{Yd}(\mathbb{G}) \subseteq \mathrm{Yd}(\mathbb{H})$. *Then if $\mathbb{C}$ codes $\mathfrak{S}$ into $\mathbb{H}$, it also codes $\mathfrak{S}$ into $\mathbb{G}$.*

Proof. Observe that $\mathbb{G} \otimes \mathbb{C} \gg \langle \mathbb{T}, \ell \otimes \mu \rangle$ iff $\mathbb{G} \gg \langle \mathbb{T}, \ell \rangle$, $\mathbb{C} \gg \langle \mathbb{T}, \mu \rangle$ and all nodes x satisfy the equations EQ. Hence $\mathbb{G} \otimes \mathbb{C} \gg \langle \mathbb{T}, \ell \otimes \mu \rangle$ iff $\mathbb{H} \otimes \mathbb{C} \gg \langle \mathbb{T}, \ell \otimes \mu \rangle$ and $\mathbb{G} \gg \langle \mathbb{T}, \ell \rangle$ iff $\langle \mathbb{T}, \ell \rangle \in \mathrm{Yd}(\mathbb{G})$ and $\langle \mathbb{T}, \ell, [[\mathsf{A}_1]], \dots, [[\mathsf{A}_n]] \rangle \in \mathfrak{S}$, and that had to be proved. $\dashv$

Hence if there is a grammar that generates all F-trees, it would be enough to code $\mathfrak{S}$ into this grammar. Again, there is a problem with the restriction to finite types. For each finite type there exists indeed such a grammar, the so-called **set-grammar** $\mathbb{S}_T(F)$. Notice that if $F = \{\mathsf{F}_1, \dots, \mathsf{F}_k\}$ then

$$\mathbb{S}_T(F) \cong \mathbb{S}_T(\mathsf{F}_1) \otimes \dots \otimes \mathbb{S}_T(\mathsf{F}_k)$$

Again, to avoid carrying around the finite type, we consider instead the grammar $\mathbb{S}^{\omega}(F)$ which contains simply all the rules of the finite subtype grammars. This grammar is again a tensor product of $1^{\sharp}$-grammars.

Theorem 7.6.4 *Let $\mathfrak{S}$ be an F-spread. Then $\mathfrak{S}$ is directly (memory) codable iff it is (memory) codable into $\mathbb{S}^{\omega}(F)$ iff some extension of $\mathfrak{S}$ seen as a label-extension is the yield of a context-free grammar.* $\dashv$

Consequently, instead of giving a code we might as well write a grammar that produces the labelled version of the extension of $\mathfrak{S}$ in order to prove direct codability.

7.7 The Coding Theorem

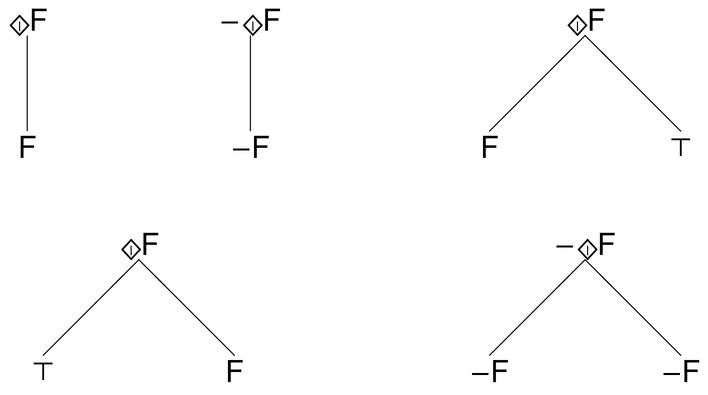
The coding theorem states that any **Ol**-specifiable spread is codable, though not necessarily memory free. This theorem is proved in several steps, simply by induction on the formation of the defining formulae and so rather than proving that such spreads can be coded, we also give direct codes in the technical sense. From now on we will also forget to talk about spreads; rather, we use the specifications instead. Moreover, most of the time we can use abstract sets rather than spreads. Finally, as we will prove direct codability, it will be enough to show codability in the easiest possible grammars, namely slot or set grammars. This basically means that we can ignore the equations in the code and just concentrate on the grammar. We begin by showing that the elementary modalities are all directly codable and proceed by a demonstration that abstract *n*-sets are codable.

Lemma 7.7.1 *$X \leftrightarrow \langle F_1?\rangle F_2$ is codable.*

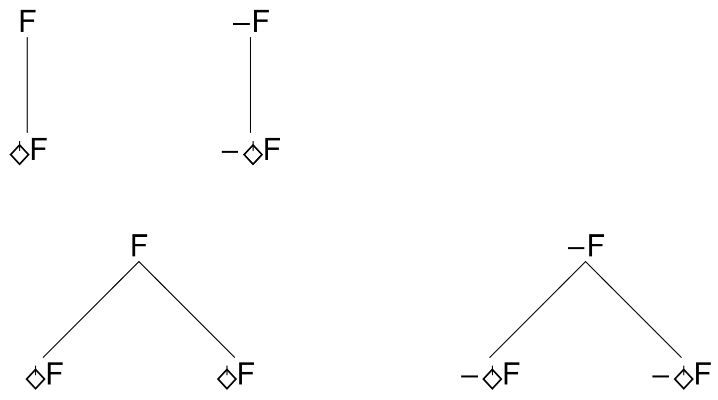
Proof. By axioms of dynamic logic $\langle F_1?\rangle F_2$ and $F_1 \cap F_2$ are the same. ⊣

Lemma 7.7.2 *$X \leftrightarrow \Diamond_{\downarrow} F$, $X \leftrightarrow \Diamond_{\uparrow} F$, $F \leftrightarrow \Diamond_{\leftarrow} F$ and $X \leftrightarrow \Diamond_{\rightarrow} F$ are codable.*

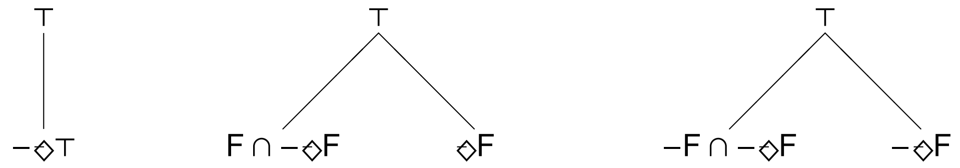
Proof. For $X \leftrightarrow \Diamond_{\downarrow} F$ consider the grammar $\mathbb{G}(\Diamond_{\downarrow})$.



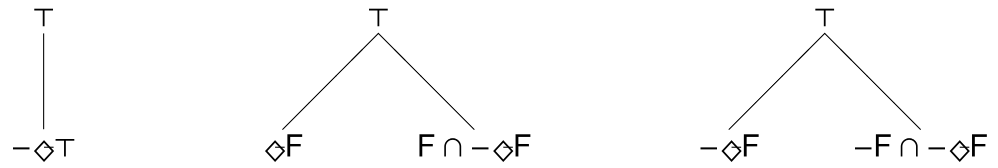
The start symbol is $\top$. By direct checking one sees that the grammar $\mathbb{G}(\Diamond_{\downarrow})$ does as promised. Now consider the grammar $\mathbb{G}(\Diamond_{\uparrow})$ with the start symbol $-\Diamond_{\uparrow}F$ (!) and the rules



Again it is directly verified that this grammar is correct for the spread $\mathsf{X} \leftrightarrow \Diamond\mathsf{F}$. Now for the grammar $\mathbb{G}(\Diamond)$ which has the start symbol $-\Diamond\mathsf{F}$ and the rules



And similarly the grammar $\mathbb{G}(\Diamond)$ with the start symbol $-\Diamond\top$ and the rules



⊣

Lemma 7.7.3 *Suppose that $X \leftrightarrow \phi(F_1, \dots, F_m)$ and $X \leftrightarrow \psi(F_1, \dots, F_m)$ is internal. Then the spreads $X \leftrightarrow \phi \cap \psi$, $X = \phi \cup \psi$ and $X \leftrightarrow -\phi$ are also internal.* ⊣

Lemma 7.7.4 *If $\mathfrak{S}_i$ specified by $X_i \leftrightarrow \phi_i(F_1, \dots, F_m)$ are codable for all $i \leq n$ then so is the n-spread $\mathfrak{S}$ defined by*

$$\begin{array}{lcl} X_1 & \leftrightarrow & \phi_1(F_1, \dots, F_m) \\ X_2 & \leftrightarrow & \phi_2(F_1, \dots, F_m) \\ \dots & & \dots \\ X_m & \leftrightarrow & \phi_n(F_1, \dots, F_m) \end{array}$$

Proof. $\mathfrak{S} = \mathfrak{S}_1 \otimes \mathfrak{S}_2 \otimes \dots \otimes \mathfrak{S}_n$. ⊣

Lemma 7.7.5 (Substitution) *Suppose that $X \leftrightarrow \phi(F_1, \dots, F_m)$ and $Y \leftrightarrow \psi(G_1, \dots, G_n)$ are codable. Then $X \leftrightarrow \phi(F_1, \dots, F_{m-1}, \psi(G_1, \dots, G_n))$ is codable.*

Proof. Let $\mathfrak{S}$ be specified by $\phi(\mathsf{F}_1, \dots, \mathsf{F}_m)$ and $\mathfrak{T}$ be specified by $\psi(\mathsf{G}_1, \dots, \mathsf{G}_n)$. Now consider the specification

$$\begin{array}{lcl} \mathsf{X}_1 & \leftrightarrow & \phi(\mathsf{F}_1, \dots, \mathsf{F}_m). \cap .\mathsf{F}_m \leftrightarrow \mathsf{X}_2 \\ \mathsf{X}_2 & \leftrightarrow & \psi(\mathsf{G}_1, \dots, \mathsf{G}_n) \end{array}$$

Let $\mathbb{G}$ code $\mathfrak{S}$ and $\mathbb{H}$ code $\mathfrak{T}$. Then $\mathbb{G} \otimes \mathbb{H}$ codes this new spread. However, the map ζ interpreting the symbols needs to be revised according to the substitution, i. e. $\zeta(X_2) = \epsilon(\mathsf{F}_m)$. ⊣

Warning. The labels G_i need not be distinct from the X_j!

Lemma 7.7.6 *If $X \leftrightarrow \langle\pi_1\rangle F$ and $Y \leftrightarrow \langle\pi_2\rangle G$ are codable then so are $X \leftrightarrow \langle\pi_1 \cup \pi_2\rangle F$, $X \leftrightarrow \langle\pi_1;\pi_2\rangle F$ and $X \leftrightarrow \langle\pi_1^+\rangle F$.*

Proof. Since $\langle\pi_1 \cup \pi_2\rangle\mathsf{F}$ is equivalent to $\langle\pi_1\rangle\mathsf{F}. \cup .\langle\pi_2\rangle\mathsf{F}$ the first result follows from boolean closure. The second follows from the Substitution Lemma; simply substitute $\langle\pi_2\rangle\mathsf{G}$ for F and we obtain $\mathsf{X} \leftrightarrow \langle\pi_1\rangle(\langle\pi_2\rangle\mathsf{G})$. Swapping F for G and observing the standard equivalence of $\langle\pi_1\rangle(\langle\pi_2\rangle p)$ with $\langle\pi_1;\pi_2\rangle p$ we get the desired result. For the last assertion let $\mathbb{H}$ code $\mathsf{X} \leftrightarrow \langle\pi_1\rangle\mathsf{F}$ where $\mathbb{G} = \mathbb{H}(\mathsf{X},\mathsf{F})$. Consider the grammar $\mathbb{M} = \mathbb{H}(\mathsf{X},\mathsf{F}\cup\mathsf{X})$. This grammar distributes X according to $\mathsf{X} \leftrightarrow \langle\pi_1\rangle(\mathsf{X}\cup\mathsf{F})$, by Substitution. This equation can be solved; it then turns into $\mathsf{X} \leftrightarrow \langle\pi_1^+\rangle\mathsf{F}$. ⊣

Theorem 7.7.7 (Coding Theorem) *A spread is memory codable iff it is **Ol**-specifiable in any set of trees of finite type.*

Proof. Suppose that $\mathfrak{S}$ is memory codable and let the type of trees considered be T. Then it suffices to consider its code in the slot grammar of type T. This code is a grammar $\mathbb{H}$. By the First Characterization we know that $\mathbb{H}$ satisfies a **Ol**-axiom $\phi(\mathsf{X}_1,\dots,\mathsf{X}_n,\mathsf{F}_1,\dots,\mathsf{F}_n)$. Then the following characterization of the spread can be given

$$\begin{array}{lcl}
\mathsf{X}_1 & \leftrightarrow & \phi_1 \cap \mathsf{X}_1 \\
\mathsf{X}_2 & \leftrightarrow & \phi_2 \cap \mathsf{X}_2 \\
\dots & & \dots \\
\mathsf{X}_n & \leftrightarrow & \phi_n \cap \mathsf{X}_n
\end{array}$$

For the converse direction, observe that by the Substitution Lemma it suffices to code

$$\begin{array}{lcl}
\mathsf{X}_1 & \leftrightarrow & \phi_1(\mathsf{F}_1,\dots,\mathsf{F}_m,\mathsf{G}_1,\dots\mathsf{G}_n) \\
\mathsf{X}_2 & \leftrightarrow & \phi_2(\mathsf{F}_1,\dots,\mathsf{F}_m,\mathsf{G}_1,\dots\mathsf{G}_n) \\
\dots & & \dots \\
\mathsf{X}_n & \leftrightarrow & \phi_n(\mathsf{F}_1,\dots,\mathsf{F}_m,\mathsf{G}_1,\dots\mathsf{G}_n)
\end{array}$$

and then substitute X_i for G_i on the right hand side. But the coding of this spread can be built up inductively with the formulas ϕ_i by the previous lemmata. ⊣

Corollary 7.7.8 *A spread is memory codable into $\mathbb{G}$ iff its restriction to the yield of $\mathbb{G}$ is Ol-specifiable.*

Proof. Again, due to the First Characterization, it is necessary that the spread be **Ol**-specifiable over the yield of $\mathbb{G}$, because if it is codable by $\mathbb{C}$ then the yield of $\mathbb{G}\otimes\mathbb{C}$ is **Ol**-specifiable. It is also sufficient by the Coding Theorem. ⊣

An important consequence is also a solution of the problem of constraint implementation. It consists in the following problem.

> *Given an F-grammar $\mathbb{S}$ and an F-sentence ϕ. Can we write a grammar that generates exactly the trees generable by $\mathbb{S}$ which satisfy the constraint ϕ? If so, ϕ is said to be* ***implementable*** *into $\mathbb{S}$.*

The constraint implementation can solved by introducing a 1-spread specified by $\mathsf{X}. \leftrightarrow .\phi$. Once this spread is coded, we cut $\mathsf{X}. \leftrightarrow .\bot$.

Theorem 7.7.9 (Constraint Coding) *A constraint on the yield of a grammar $\mathbb{S}$ is implementable iff it is* **Olt***-specifiable over the yield of $\mathbb{S}$.* ⊣

The actual size of the memory needed to code a spread is not easy to establish. This is so because the number of features needed does not match one-by-one the inductive steps of the proof. Subtle methods are called for. The problem we pose is the question of *constraint implementation*. Obviously, the set of subformulas of ϕ gives an indication of the size; but this can only be a rough upper bound. There are two differences to be accounted for. One is that horizontal modalities if stacked onto each other introduce no extra cost; $\Diamond\Diamond\mathsf{D}$ is as cheap as $\Diamond\mathsf{D}$. The other is that $\lozenge^*\mathsf{D}$ is as expensive as $\lozenge\mathsf{D}$. To make the best use of these facts we introduce the function σ, which selects from the so-called *Fischer-Ladner* closure those subformulas which introduce an extra memory. The proper definition is quite lengthy. Notice that σ returns for given ϕ a set of formulas of ϕ. For the understanding of the definition let $\lozenge$, $\blacklozenge$ range over programs, and let a program be called *horizontal* if it is free of $\overset{\shortmid}{\Diamond}$, $\underset{\shortmid}{\Diamond}$ (though it may contain tests, stars etc.).

$$\begin{array}{lcll}
\sigma(\mathsf{F}) & = & \emptyset & \text{if } \mathsf{F} \in F \\
\sigma(\mathsf{F}) & = & \{\mathsf{F}\} & \text{if } \mathsf{F} \notin F \\
\sigma(-\phi) & = & \sigma(\phi) & \\
\sigma(\phi \wedge \psi) & = & \sigma(\phi) \cup \sigma(\psi) & \\
\sigma(\phi \vee \psi) & = & \sigma(\phi) \cup \sigma(\psi) & \\
\sigma(\langle \phi ? \rangle \psi) & = & \sigma(\phi) \cup \sigma(\psi) & \\
\sigma(\lozenge\phi) & = & \{\lozenge\phi\} \cup \sigma(\phi) & \lozenge \text{ horizontal} \\
\sigma(\overset{\shortmid}{\Diamond}\phi) & = & \{\overset{\shortmid}{\Diamond}\phi\} \cup \sigma(\phi) & \\
\sigma(\underset{\shortmid}{\Diamond}\phi) & = & \{\underset{\shortmid}{\Diamond}\phi\} \cup \sigma(\phi) & \\
\sigma((\lozenge \cup \blacklozenge)\phi) & = & \sigma(\lozenge\phi) \cup \sigma(\blacklozenge\psi) & \\
\sigma((\lozenge \circ \blacklozenge)\phi) & = & \sigma(\lozenge(\blacklozenge\psi)) & \\
\sigma((\lozenge \cup \blacklozenge)^*\phi) & = & \sigma(\lozenge(\lozenge \cup \blacklozenge)^*\phi) \cup \sigma(\blacklozenge(\lozenge \cup \blacklozenge)^*\phi) \cup \sigma(\phi) & \\
\sigma((\lozenge \circ \blacklozenge)^*\phi) & = & \sigma(\lozenge \circ \blacklozenge \circ (\lozenge \circ \blacklozenge)^*\phi) \cup \sigma(\phi) & \\
\sigma((\lozenge)^{**}\phi) & = & \sigma(\lozenge^*\phi) &
\end{array}$$

First of all, one needs to see that σ is well-defined. The boolean connectives are covered. So let us take a formula $\lozenge\phi$. If $\lozenge$ is basic, we have either the clause for horizontal programs, or the extra clauses for $\Diamond$, $\Diamond$. The cases $\Diamond = \Diamond_1 \circ \Diamond_2$, $\Diamond = \Diamond_1 \cup \Diamond_2$, and the test are covered as well. This leaves the case $\Diamond = \blacklozenge^*$. Only the subcases of union and composition are critical. We wave a rigorous proof here of the fact that the definition of σ is well-founded, but such a proof can be given.

Theorem 7.7.10 *The constraint ϕ can be coded with a memory of at most $\sharp(\sigma(P))$.*

Proof. It suffices to show that if the elements of $\sigma(P)$ are all coded, so is P, and that all elements are codable in a single step from some other elements in $\sigma(P)$. For example, since $\blacklozenge^{**}\phi. \leftrightarrow .\blacklozenge^*\phi$, the first is coded if the latter is. The code for $(\blacklozenge \cup \Diamond)^*\phi$ arises from the specification $\mathsf{X}. \leftrightarrow .\phi \cup (\Diamond \cup \blacklozenge)\mathsf{X}$. To code that we need to code ϕ. Furthermore, we need to code $\Diamond\mathsf{X}$ and $\Diamond\mathsf{X}$, which turn out to be equivalent to $\Diamond(\blacklozenge \cup \Diamond)^*\phi$ and $\blacklozenge(\blacklozenge \cup \Diamond)^*\phi$. (Actually, the procedure is first to code $\Diamond\phi$ and $\blacklozenge\phi$ and then to replace that code by the fixpoint determined by X.) And so on. $\dashv$

Chapter 8

Nearness Grammars

8.1 Motivation and Definition

The Coding Theorem has the advantage to calibrate exactly the codable spreads, but does not correspond to the way linguists normally think about syntax. Therefore we need to liberate ourselves step by step from the low level Coding Theorem by developing higher order tools. One is the notion of a *nearness grammar*. A nearness grammar allows to specify directly certain functional dependencies between elements in a tree which need not be close. This allows to relax the local structure of a cfg considerable. Remember a cfg is a grammar in which the only accessible information is the one that resides in the label of a node and two nodes can influence each other only if they are close. Yet, the influence spheres of elements are far typically larger than that. In order to be able to state directly which nodes are influenced in what way by a node, we present the following definition. Let p be a nearness term and A, B be two labels. We define three types of nearness conditions based on p, A and B.

- Existential $\mathcal{E}(\mathsf{A},\mathsf{p},\mathsf{B})$
- Universal $\mathcal{U}(\mathsf{A},\mathsf{p},\mathsf{B})$
- Definite $\mathcal{D}(\mathsf{A},\mathsf{p},\mathsf{B})$

A tree $\mathfrak{T}$ satisfies the existential nearness condition $\mathcal{E}(\mathsf{A},\mathsf{p},\mathsf{B})$ iff for all $x \in T$ with label A there is a $y \in xp$ of label B. $\mathfrak{T}$ satisfies the universal nearness condition $\mathcal{U}(\mathsf{A},\mathsf{p},\mathsf{B})$ iff for all $x \in T$ of label A and all $y \in xp$ y is of label B. Finally, $\mathfrak{T}$ satisfies the definite nearness condition $\mathcal{D}(\mathsf{A},\mathsf{p},\mathsf{B})$ iff for all $x \in T$ there is exactly one $y \in p$ of label B.

Definition 8.1.1 *A* ***nearness grammar*** *over* $\mathfrak{L}$ *is a pair* $\mathbb{N} = \langle \mathbb{G}, \mathfrak{N} \rangle$*, where* $\mathbb{G}$ *is a boolean grammar and* $\mathfrak{N}$ *is a finite set of nearness conditions.* $\mathbb{G}$ *is called the* ***generative component*** *of* $\mathbb{N}$ *and* $\mathfrak{N}$ *the* ***nearness filter****.*

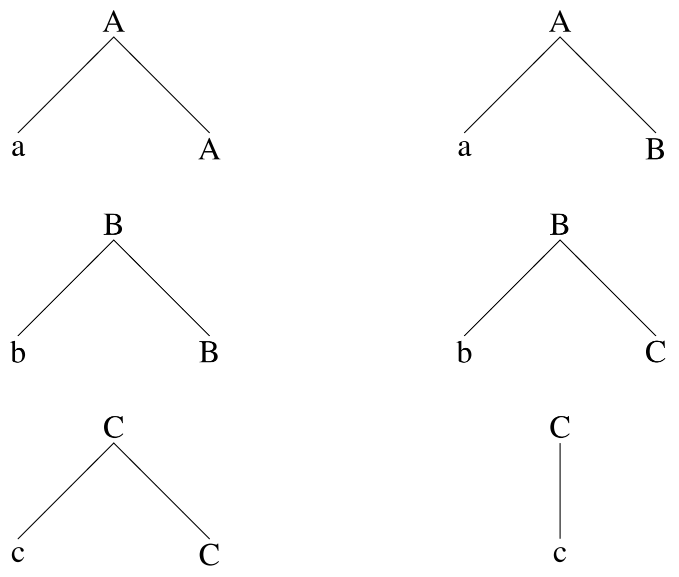
With $\mathbb{N} = \langle \mathbb{G}, \mathfrak{N} \rangle$ a nearness grammar we write $\mathbb{N} \gg \mathfrak{T}$ if

(Generative Condition)	$\mathbb{G} \gg \mathfrak{T}$
(Nearness Filter)	$\mathfrak{T}$ satisfies all $\mathfrak{n} \in \mathfrak{N}$

The complexity of nearness grammars is measured by two ordinary complexity measures: one for the generative component and one for the nearness filter. For example, we call $\mathbb{N}$ *context-free with regular (nearness) filter* if the generative component is context-free and the ovals of the nearness filter are all regular. We say that $\mathbb{N}$ is **context-free** if it is equivalent to a context-free grammar in the sense that there is a context-free grammar generating exactly the same labelled trees. It is quite intriguing to see what effect nearness conditions have if unravelled into plain phrase-structure rules. One main result of this work will be that as long as the nearness filter is *regular* it can be encoded into the phrase-structure rules without affecting the complexity of the grammar (except for the number of labels which of course increases). A regular grammar with regular filter is again regular, a linear grammar with regular filter is again linear and a context-free grammar with regular filter is context-free. However, if we go beyond regularity in the filter, nothing can be guaranteed.

8.2 A Pathological Example

We will show that the complexity of the nearness filter affects the the total complexity of the grammar rather drastically if it is not regular. Our particular example is the following grammar.



This grammar, called $\mathbb{G}$, is right linear and generates the regular language $a^+b^+c^+$. Consider now the following nearness conditions. Let

$$\mathbb{O}_l = \langle A|\{A^m B^{m+n-1} C^n | 1 \leq m, n \in \omega\}\rangle$$

$$\mathbb{O}_r = \langle \{C^m B^{m+n-1} A^n | 1 \leq m, n \in \omega\}|A\rangle$$

Lemma 8.2.1 *$\mathbb{O}_l$ and $\mathbb{O}_r$ are context-free.*

Proof. We have to show that $\{A^m B^{m+n-1} C^n | m, n \in \omega\}$ and its transpose are context-free. To prove this for one of them is enough. Notice namely, that this language is a union of $A \cdot L_1 \cdot L_2$ and $L_1 \cdot L_2 \cdot C$ where $L_1 = \{A^m B^m | m \in \omega\}$ and $L_2 = \{B^n C^n | n \in \omega\}$, which both are context-free. Context-free languages are closed under concatenation and union. ⊣

Now we investigate the effect of adding $\mathcal{E}(a, \mathbb{O}_l, c)$ or $\mathcal{E}(c, \mathbb{O}_r, a)$ to the grammar $\mathbb{G}$.

Lemma 8.2.2 *$\mathbb{N} = \langle \mathbb{G}, \{\mathcal{E}(a, \mathbb{O}_l, c), \mathcal{E}(c, \mathbb{O}_r, a)\}\rangle$ generates the language*

$$\{a^m b^m c^m | 1 \leq m \in \omega\}$$

Proof. Consider the string $a^r b^s c^t = a \ldots ab \ldots bc \ldots c$. There is exactly one way to parse this string in $\mathbb{G}$ namely as

$$[_A\, a[_A\, a \ldots [_A\, a[_B\, b[_B\, b \ldots [_B\, b[_C\, c[_C\, c \ldots [_C\, c] \ldots]]] \ldots]]] \ldots]]$$

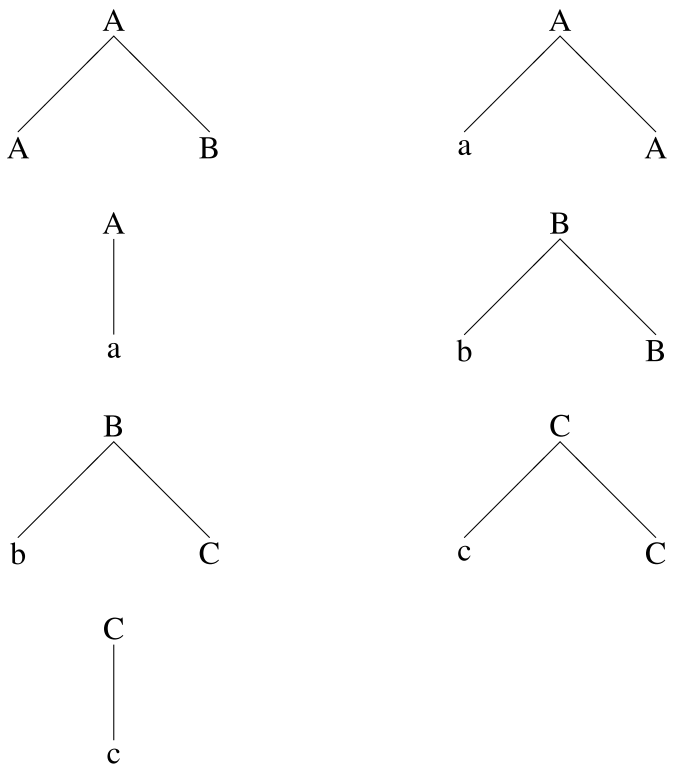
Now observe a few things. Each a corresponds to a unique A covering it, and each A corresponds to a unique a covered by it, similarly the b and the B and the c and the C are connected. Now take the string $a^r b^s c^t$ and consider the last a. We know that for some m, n, there must be a c such that $c \in a\langle A|A^m B^{m+n-1} C^n\rangle$. Clearly, $m = 1$ so that we have $c \in a\langle A|AB^n C^n\rangle$, for some $n \geq 1$. It follows in this case that $s = n$ and $t \geq n$ and so $t \geq s$. Take then the last c. Since $a \in c\langle C^m B^{m+n-1} A^n|A\rangle$, we must have $m = t$, $m + n - 1 = s$ and so $n = s - m + 1 = s - t + 1$. From $n \geq 1$ we deduce that $s - t \geq 0$ and so $s \geq t$. Together this gives $s = t$. Consider then the first c in the string. According to the nearness filter, there must be an $a \in c\langle C^m B^{m+n-1} A^n|A\rangle$ for some m, n. By choice of the c, $a \in c\langle CB^n A^n|A\rangle$ for some $n \geq 1$. Hence, $n = s$ and $r \geq n$, from which we deduce $r \geq s$. Now consider the first a. Since $c \in a\langle A|A^m B^{m+n-1} C^n\rangle$ for some $m, n \geq 1$ we have in this constellation $m = r$ and $s = m + n - 1$ and thus $n = s - m + 1 = s - r + 1$. From $n \geq 1$ we have $s - r \geq 0$ whence $s \geq r$. Together we have $s = r$. ⊣

Theorem 8.2.3 *There are right linear grammars and existential context-free nearness filters which give rise to a non-context free language.* ⊣

We could have proved this result taking just one of the two nearness conditions, but the result would not have been that convincing. Notice that this result generalizes easily. It is not hard to generate any language

$$\{a_1^n a_2^n \dots a_m^n | n \in \omega\}$$

from a similar right linear grammar and a set of nearness conditions. Notice finally that the effect of the filter imposed by nearness conditions varies with the generative component. If we had chosen to generate $a^+b^+c^+$ by this grammar instead then the language generated by this grammar together with the nearness conditions given above is empty.



8.3 Codability of Nearness Restrictions

Theorem 8.3.1 *A nearness grammar with context free generative component and regular filter is context free.*

This theorem will be proved by translating nearness conditions into spread specifications. To this end consider a regular nearness polynomial p. We will translate this polynomial into a regular modality as follows.

$$\begin{array}{lcl}
\langle \mathsf{A}|^\diamond & = & \overset{|}{\Diamond} \circ \langle \mathsf{A}?\rangle \\
|\mathsf{A}\rangle^\diamond & = & \langle \mathsf{A}?\rangle \circ \underset{|}{\Diamond} \\
(p \cdot q)^\diamond & = & p^\diamond \circ q^\diamond \\
(p \cup q)^\diamond & = & p^\diamond \cup q^\diamond \\
(p^*)^\diamond & = & (p^\diamond)^*
\end{array}$$

The last equation has to be taken cum grano salis because the star is not really defined on p but rather on path-sets; but that in effect means that only the left hand side is restricted, and the translation works in all cases. Furthermore, we let $p^\Box\mathsf{A}. \leftrightarrow\ . - p^\diamond - \mathsf{A}$.

Lemma 8.3.2 *$y \in xp$ iff $xp^\diamond y$.*

Proof. By induction. ⊣

Observe that for regular nearness polynomials p the translation is regular in $\underset{|}{\Diamond}$, $\overset{|}{\Diamond}$, $-\!\Diamond$, $\Diamond\!-$ and the tests $\langle \mathsf{A}?\rangle$. Consequently, we can define the converse $(p^\diamond)^\smile$, which is then a regular modality as well.

Lemma 8.3.3 *A labelled tree $\mathfrak{T}$ satisfies $\mathcal{E}(\mathsf{A}, p, \mathsf{B})$ iff it satisfies the axiom $\mathsf{A} \to p^\diamond\mathsf{B}$. $\mathfrak{T}$ satisfies $\mathcal{U}(\mathsf{A}, p, \mathsf{B})$ iff it satisfies the axiom $\mathsf{A} \to p^\Box\mathsf{B}$.*

Proof. $\mathfrak{T}$ satisfies $\mathsf{A} \to p^\diamond\mathsf{B}$ iff for all nodes x of label A there is a y such that $xp^\diamond y$ and y is of label B iff for all x of label A there is a $y \in xp$ of label B iff $\mathfrak{T}$ satisfies $\mathcal{E}(\mathsf{A}, p, \mathsf{B})$. Similarly for $\mathcal{U}(\mathsf{A}, p, \mathsf{B})$. ⊣

These two conditions are comparatively easy to handle. For the definite nearness condition we have to do some more work. It can be simplified by breaking the uniqueness condition into two parts. If namely for an A-node there is a unique B-node, then there also is a unique crosspoint. Moreover, if we have that every A is connected with a unique crosspoint and the crosspoint is itself connected with a unique B-node, then the B-node connected with the A-node is unique. So let us be given an oval $\mathbb{O} = \langle L|R\rangle$, and let $\mathbb{O}_\ell = \langle L|$, $\mathbb{O}_r = |R\rangle$. Introduce a new feature X into grammar and replace $\mathcal{D}(\mathsf{A}, \mathbb{O}, \mathsf{B})$ by the two conditions $\mathcal{D}(\mathsf{A}, \mathbb{O}_\ell, \mathsf{X})$ and $\mathcal{D}(\mathsf{X}, \mathbb{O}_r, \mathsf{B})$. If the latter two are satisfied, so is the first; and conversely. Similar reductions can be made wit respect to n-spheres. Consequently, we can restrict ourselves to discussing 1-spheres only. For example, $\mathcal{D}(\mathsf{A}, \langle L|, \mathsf{X})$. Let A be an accepting automaton for L with intial state i_0 and accepting states F. For any pair i, j of states write $[i : j]$ for the set of all strings that lead A from state i to state j. This is obviously a regular language for any choice of i and j. The key idea is now this. Starting from an A-node x we inititialize A and travel up the tree. As soon as A comes into an accepting state $f \in F$ and we hit a X-node y, there may not be any further X-nodes on an accepting state. This requirement can be recast in the orientation language. Let $[i : j]^\diamond$ abbreviate $\langle [i : j] |^\diamond$ and $[i : j]^\Box$ abbreviate $-[i : j]^\diamond -$. Then consider the following postulate.

$$\mathsf{A}. \to . \bigwedge_{f \in F} [i_0 : f]^\Box (\mathsf{X}. \to . \bigwedge_{f' \in F} [(f : f') - \epsilon]^\Box - \mathsf{X})$$

It says that on condition we are at an A-node, and on travelling upwards we meet an X-node y in state f, then any different (!) node above which is reached in an accepting state must be a $-$X-node. Notice that we have to chose $[(f : f') - \epsilon]$ rather than $[(f : f') - \epsilon]$ to avoid quantifying over the crosspoint as well. If we want to code the downward looking uniqueness restriction, we just have to exchange the upward looking $\langle [i : j] |$ by the downward looking $|[i : j]\rangle$ throughout. The proof runs completely analogous. This finishes the proof of Theorem 8.3.1. ⊣Obviously, one can think of more nearness filters; for example, we can require an A-node to have a fixed amount of B-nodes within it's influence sphere. Little reflection is needed, that for elementarily definable generalized quantifiers Q the nearness filter $Q(\mathsf{A}, p, \mathsf{B})$ is codable. But we will not dwell on this issue.

8.4 Schematic Nearness Grammars

For practical purposes, the concept of a nearness grammar is too low level. In the following two paragraphs we will therefore provide two ways of generalizing this concept to make it more useful for practical linguistic work. Moreover, we will derive a rather general theorem which we call the *A Priori Theorem* which determines whether particular grammars exceed the limits of context freeness.

The main deficiency of nearness grammars is that the obvious notion of passing on information from one node to another is not directly expressed. For example, if we want to express that a trace of category C and bar-level ℓ needs an antecedent of category C and bar level ℓ, we must specify that for each possible choice of C and ℓ. So we need to state that an N-trace needs an N-antecedent, a $\overline{\overline{\mathsf{N}}}$-trace needs a $\overline{\overline{\mathsf{N}}}$-antecedent, a V needs a V-antecedent etc. This is not an ideal situation, not only because this procedure is rather boring. It also misses the point of the dependeny; the obvious general form which this dependency has is not mirrored in the code. Therefore linguists have adopted the use of schematic variables. In HPSG with its AVM formalism one would replace the list of dependencies by just one:

$$\mathcal{E}(\begin{bmatrix} \text{TRACE}{:}\mathit{yes} \\ \text{CAT} \ : \ x \\ \text{BAR} \ : \ \lambda \end{bmatrix}, p, \begin{bmatrix} \text{CAT}{:}x \\ \text{BAR}{:}\lambda \end{bmatrix})$$

x and λ are allowed to range over all possible values. Let us call a nearness grammar *schematic* if in addition to concrete categories schematic variables are used in the nearness filters. Obviously, this step is critical because we do not know whether the range of the variables is finite. However, if it is, the schematic nearness filters can be replaced by ordinary nearness filters and we derive what we call the HPSG-variant of the

Theorem 8.4.1 (A Priori: HPSG) *A schematic nearness grammar with context free generating component and regular nearness filters is context free if the schematic variables can only range over finitely many values.* ⊣

This part of the theorem is actually quite easy although generally quite useful. The other, the GB half, is more demanding. If, namely, the schematic variables cannot a priori be said to range over a finite set of values, is there nevertheless a criterion for context freeness? Let us be concrete and take again movement as an example. A first glance at formulations of movement suggest that as above each trace of certain category has a unique antecedent of identical category, and that the indices are just for better readability. But that turns out to be false on close inspection. Especially head-movement – about which we will have something to say later on – cannot be formalized without the use of indices. So the scheme above needs to include a variable for an index. With a nearness polynomial to be fixed by an individual theory, the overall scheme for movement in GB is this

$$\mathcal{D}(\begin{bmatrix} \text{TRACE}: yes \\ \text{CAT} \;:\; x \\ \text{BAR} \;:\; \lambda \\ \text{IND} \;:\; i \end{bmatrix}, p, \begin{bmatrix} \text{CAT}: x \\ \text{BAR}: \lambda \\ \text{IND}: i \end{bmatrix})$$

The range of i is the set of natural numbers, so there is no hope of using the a priori argument in this form. Rather, one needs to investigate the role of p quite closely. The key is the Dependency Lemma. It states that a dependency from x to y needs to involve all nodes of $dist(x, y)$, without the crosspoint, however. Given p we define an *extraction path* of p to a trace without it's maximal and minimal points unless they are start and end point. So, from x we trace p, going up and down etc. until we arrive at y. Intermediate extreme points are omitted. For example, in the normal case p is an oval and the extraction path from x to y is $\uparrow x \cap \uparrow y - \{z\}$, where z is the crosspoint. Such paths, even though defined not quite the same way, are essential in certain theories of movement, such as Pesetzky's. The extraction paths give us the exact set of points needed to control the fulfillment the nearness filter. For the sake of illustration think of a fire brigade getting water from a lake to the spot where a house is burning; the men have to line up quite close to each other in order to be able to pass on the buckets. Likewise in syntactic trees. The nodes which are supposed to transport information have to be *close* – in the technical sense. The Independence Lemma says that if they are not, the information chain is broken. So, selecting a minimal chain of close nodes linking x to y is necessary and sufficient. This suffices for ovals, but fails in the general case, because of lack of EMBEDDABILITY. Here, it is important to follow the attempted voyages from x of the polynomial p in order to find out whether or not we reach y.

Along each extraction path the information about the trace needs to be stored together with information about the relative position of the actual node in the (attempted) voyage. If the polynomial is regular, the information about the position is finite. The trace, however, has an index. Yet indices are a tricky thing. The actual number of the index is unimportant, it can be randomly assigned as long as certain identity criteria and non-identity criteria are observed. Indices are therefore unobservables. Howevere, indices seem to be essential to GB because they make the formulation of movement at all possible. Without indices there might be several antecedents within the influence sphere in question, even though it is desired to have a one-to-one correspondence between traces and antecedents within their respective spheres. Now consider the case where there is an a priori bound on the number of extraction paths having a node in common. Call this number E. Then it seems that rather than taking all numbers as indices, we only need some finite set of numbers. This can be done, but needs some care. In fact, we assign to each node a number as *trace* and a number as *antecedent*. (In case something is not a trace, the trace number is arbitrary, likewise if something is not an antecedent.) So, we have $E \times E$ many indices. Each individual node in the tree just needs to keep track of E many traces with their respective distance. A trace just has to select an index which is not at present used. It can always do so, because it knows which numbers are already given away. There must be such a number because at the node of the trace there can be at most E coexisting extraction paths, one being the path of the trace itself. We conclude

Theorem 8.4.2 (A Priori: GB) *A context free grammar with schematic nearness filter is context free if the filter is regular and there exists an a priori bound on the number of overlapping extraction paths.* $\dashv$

This theorem is highly significant. (Manzini, 1992) states explicitly that her theory predicts there can be no more than two overlapping $\overline{A}$-extraction paths. Her extraction paths coincide with ours, so it is immediate that *locality* for $\overline{A}$-movement is context free.

Finally a word on memory. If we have a nearness condition $\mathcal{D}(\mathsf{A}, p, \mathsf{B})$ or any other type, then A and B are standardly assumed to be accessible symbols, i. e. composed from the feature set. The code of the nearness condition works by pairing the symbols A, B with a distance tag that give information as to how far the corresponding syntactic item is that we wish to monitor. The tag comes from a finite set T and depends only on the distance polynomial and the quantifier of the nearness condition. It does not depend on A and B. This has the consequence that the memory needed to code a schematic nearness condition $\mathcal{D}(\mathsf{A}[\mathsf{X}], p, \mathsf{B}[\mathsf{X}])$ is $\leq {}^2log\sharp T$, the number needed to create $\sharp T$ atoms in a boolean algebra. So, an infinite labelling mereology is expanded only by a finite set of features in order to code a schematic nearness condition. However, on the level of atoms the situation is different; the number (=cardinality) of atoms is multiplied by T. Here we see the advantage in the somewhat roundabout definition of a memory. In the infinite case it would be hard to quantify the memory otherwise. For example, $\aleph_0 \cdot \sharp T = \aleph_0$, so one could by a suitable code do with no more memory.

8.5 Functional Dependencies

Not always can a dependency be factored out so nicely with the introduction of variables. An interesting case is morphological agreement. In HPSG it is argued that GB is overly redundant with it's interpretation of agreement. The principal reason is that the information flow is assumed to be one way in GB rather than two way as in HPSG. That this should matter is exemplified with particular redundancies in agreement. German, with it's particularly rich abstract inflectional system, having three genders, four cases, singular/plural and three inflectional paradigms (weak/strong/mixed), allows in principle for 72 different form per lexeme. In fact, adjectives only have fourteen different forms, and the abstract classification allows for more distinctions than actually present in the morphology. Nouns obviously have even less forms because they have fixed gender. Different classes of lexemes cut the space of 72 forms in different ways. Two issues arise. The first is, whether the whole classification is adequate for German inflectional morphology. Much has been said on this and it is not the purpose of this book to comment on it. We assume here that the classification is correct and optimal. The second question is what to do with the inflectional covariance in face of the fact that lexeme classes show different redundancies. To see the point, look at the following example.

(MS)	Je suis heureux.
(FS)	Je suis heureuse.
(MP)	Nous sommes heureux.
(FP)	Nous sommes heureuses.

The subject noun is ambiguous with respect to gender while the adjective is not. Both are unambiguous with respect to number, but in the phonetics the adjective is in fact ambiguous. GB assumes that the noun takes the lead and sets the agreement features (masculine/feminine), (singular/plural), and passes the values on to the dependent elements. While this underlying process generates all theoretically necessary distinctions, the actual sentence will not display certain choices, whence the ambiguity. Notice that with respect to gender we can just as well say that there is no depedency from adjective to subject, at least if we look at the surface forms. We could namely set the subject to *masculine* and nevertheless generate a feminine adjective. The mismatch is not observable at the surface string. HPSG takes notice of this fact and lets the subject be underdetermined with respect to gender. The adjective takes the lead instead with respect to gender agreement features, and lets the information flow the other way around. One of arguments adduced in favour of this procedure is that it is more economical than a one-way information flow. But this argument has not much more than rhetorical force. First of all, the difference in the two approaches lies principally in the difference with respect to analysing language from the parsing point of view as opposed to analysing it from the generating point of view. If I *utter* (MS), then it is perfectly sensible to say that the gender value is fixed at the subject, even though that is not at all visible. If I *hear* (MS), on the other hand, I must defer a decision on gender until a suitable point for clarification is reached. These two views present aspects of the same thing in different use, the grammar. However, the grammar as an object is insensitive as to in which direction we perceive the information to flow. A dependency from x to y can be seen as a dependency from y to x, because the two are covariant.

Still, the argument is a formal one, namely that GB is uneconomical in it's use of agreement features. Let us see how much force this argument has. We have to return to schematic variables. A priori, what GB assumes is that a schematic variable, fully instantiated, is transferred. So there is a dependency, let us say, of this form.

$$\mathcal{U}(\mathsf{A}[\mathsf{X}], p, \mathsf{B}[\mathsf{X}])$$

The concern now is that for several values of X, A[X] has identical form, and that for some choices of X, the corresponding B[X] are identical. At each end, there is thus a certain redundancy. These redundancies might differ, because the categories A and B need not match. Let us for the moment assume that the category information is not passed on. Thus, B at the receiving end of the line, does not know whether it agrees with an adjective, or a noun, or a prounoun etc. This is important for the argumentation. Without this assumption, the whole problem becomes much more complex, but the situation is in nuce the same. As for the transfer, it is not necessary for B to know the choice at A exactly, only inasmuch as it's own distinctions are concerned; and likewise for A. A brute force method would be to suggest different classification schemes for A and for B, and produce a look-up table as to which corresponds to which. This look-up table will be implemented at the crosspoint. The branch leading down to the A-point and the branch leading down to the B-point are coupled at the crosspoint, and so the grammar rules simply have to be written in such a way that each crosspoint creates the correct combinations for both branches. The information sent to A depends on the classificatiry scheme for A, and the information sent to B depends on the classificatory scheme for B.

Suppose (spoken!) *Mini-French* was the only language of the universe (so it's grammar is identical to UG) and suppose Mini-French just had sentences similar to the ones above, say a sequence *pronoun+copula+adjective*. Mini-French linguists would not posit any agreement. Neither would the Mini-French GB linguist. The distinctions of UG will in this case not go any deeper than Mini-French. Any postulated agreement, though not in itself inconceivable, will fall prey of Ockham's razor. It is also subject to the observability principle. The classification scheme for the subject is independent of the classification scheme for adjectives. Now consider *Midi-French*, a language expanding Mini-French by having also ordinary nouns and determiners. In Midi-French there is agreement, namely (if spoken sentences are concerned) only in gender. So the Midi-French linguist will posit agreement with respect to gender, not with respect to number. The problem of controlling agreement in grammar has to be solved now. One possiblity is to layer a grammar of Midi-French on top of Mini-French by implementing agreement only with respect to determiners and nouns which are not-pronouns. Finally, Maxi-French is written Midi-French, and so reveals a number distinction in adjectives as well. The Midi-French linguist will observe a distinction between pronouns and noun phrases with respect to agreement. He can either declare them to be distinct instances of a phrasal category, one with gender marking and one without. Or he can assume each pronoun to stand for two homophonic pronouns, one for each gender. Question for the grammar implementation is whether agreement knowledge needs to be transported for pronouns. We have assumed, however, that the categorial identity is *not* transferred, so that the receiving adjective, if it has to agree, must know the gender of the subject because it cannot decide whether that is relevant. Could it know it is agreeing with a pronoun, it needn't care. But to transfer such knowledge costs extra memory, i. e. multiplies the symbols of the grammar.

This is precisely the point where the argument of HPSG runs into difficulties. Even though in HPSG the question of quantity of transferred information is taken seriously, it is dificult to quantify exactly the amount of information to be transferred. Notice that our approach via coding allows to measure this much more directly than with HPSG. The reason lies in the peculiar mechanism of reentrancy. The latter, being of metaphysical significance to HPSG is a hidden source of memory explosion, because any reentrancy even though cutting the size of the representation locally does so at the cost of relaxing into a global structure – just as does GB. The price is the same; it means that when one is parsing a sentence global information has to be available at the spot, it has to local. This time it is by means of pointers to reentrant points. The predicate being able to tap the resources of agreement so nicely by reentering the feature system of the subject – this luxury means that even after succesfully parsing the subject we need to remember part of it's structure if we do not want to lose these reentrancy points. The point of our search for the cost of coding tries to quantify exactly *how much* we need to remember about the subject in order to carry on successfully.

Bibliography

(Baker, 1988) Mark C. Baker. *Incorporation. A theory of grammatical function changing*. Chicago University Press, 1988.

(Baker, 1992) Mark Baker. Unmatched chains and the representation of plural pronouns. *Journal of Semantics*, 1:33–74, 1992.

(Barker and Pullum, 1990) Chris Barker and Geoffrey Pullum. A theory of command relations. *Linguistics and Philosophy*, 13:1–34, 1990.

(Barwise and Perry, 1986) John Barwise and John Perry. *Situations and Attitudes*. MIT Press, Cambridge (Mass.), 4 edition, 1986.

(Becker and Walter, 1977) Heinrich Becker and Herrmann Walter. *Formale Sprachen*. Vieweg, Braunschweig/Wiesbaden, 1977.

(Bird and Klein, 1990) Steven Bird and Ewan Klein. Phonological events. *Journal of Linguistics*, 29, 1990.

(Blake, 1994) Barry J. Blake. *Case*. Cambridge Textbooks in Linguistics. Cambridge University Press, 1994.

(Bußmann, 1990) Hadumod Bußmann. *Lexikon der Sprachwissenschaft*. Number 452 in Kröners Taschenausgabe. Kröner, Stuttgart, 2 edition, 1990.

(Ćavar and Wilder, 1994) Damir Ćavar and Chris Wilder. Word order variation, verb movement, and economy principles. *Studia Linguistica*, 48:46 – 86, 1994.

(Chomsky, 1980) Noam Chomsky. *Rules and Representations*. Columbia University Press, 1980.

(Chomsky, 1981) Noam Chomsky. *Lecture Notes on Government and Binding*. Foris, Dordrecht, 1981.

(Chomsky, 1986a) Noam Chomsky. *Barriers*. MIT Press, Cambrigde (Mass.), 1986.

(Chomsky, 1986b) Noam Chomsky. *Knowledge of Language. Its Nature, origin and Use*. Praeger, New York, 1986.

(Chomsky, 1991) Noam Chomsky. Some notes on economy of derivations. In Robert Freidin, editor, *Principles and Parameters in Comparative Grammar*. MIT Press, Cambridge (Mass.), 1991.

(Chomsky, 1993) Noam Chomsky. A minimalist program for linguistic theory. In K. Hale and Keyser S. J., editors, *The View from Building 20: Essays in Honour of Sylvain Bromberger*, pages 1 – 52. MIT Press, 1993.

(Chomsky, 1994) Noam Chomsky. Bare phrase structure. Technical report, MIT, 1994.

(Dodds, 1980) R. W. Dodds. *Teach Yourself Malay. A complete course for beginners*. Teach Yourself Books. Hudder and Stoughton, Sevenoaks, Kent, 2 edition, 1980.

(Fanselow and Felix, 1987) Gisbert Fanselow and Sascha Felix. *Sprachtheorie, Bd. 2: Die Rektions- und Bindungstheorie*. UTB No. 1441/2. Tübingen, 1987.

(Fanselow, 1991) Gisbert Fanselow. *Minimale Syntax*. Number 32 in Groninger Arbeiten zur germanistischen Linguistik. Rijksuniversiteit Groningen, 1991.

(Fiengo and May, 1994) Robert Fiengo and Robert May. *Indices and Identity*. Number 24 in Linguistics Inquiry Monographs. MIT Press, 1994.

(Frey, 1993) Werner Frey. *Syntaktische Bedingungen für die semantische Interpretation*. Number 35 in Studia Grammatica. Akademie Verlag, Berlin, 1993.

(Gazdar *et al.*, 1985) Gerald Gazdar, Ewan Klein, Geoffrey Pullum, and Ivan Sag. *Generalized Phrase Structure Grammar*. Blackwell, Oxford, 1985.

(Gazdar *et al.*, 1988) G. Gazdar, G. K. Pullum, R. Carpenter, E. H. Klein, T. E. Hukari, and R. D. Levine. Category structures. *Journal of Computational Linguistics*, 14-1:1–14, 1988.

(Geach, 1972) Peter Geach. A Program for Syntax. In Donald Davidson and Gilbert Harman, editors, *Semantics for Natural Language*, number 40 in Synthese Library. Reidel, Dordrecht, 1972.

(Haegeman, 1991) Liliane Haegeman. *Introduction to Government and Binding Theory*. Blackwell, Oxford, 1991.

(Haider, 1992) Hubert Haider. Branching and discharge. Technical Report 23, SFB 340, Universität Stuttgart, 1992.

(Haider, 1993) Hubert Haider. *Deutsche Syntax – generativ. Vorstudien zur Theorie einer projektiven Grammatik*. Gunter Narr Verlag, Tübingen, 1993.

(Haider, 1994) Hubert Haider. Detached clauses – the later the deeper. Technical Report 41, SFB 340, Universitiät Stuttgart, 1994.

(Halle and Marantz, 1993) Morris Halle and Alec Marantz. Distributed morphology and the pieces of inflection. In K. Hale and Keyser S. J., editors, *The View from Building 20: Essays in Honour of Sylvain Bromberger*, pages 111 –176. MIT Press, 1993.

(Happa, 1975) Heinz Happa. *Grundfragen einer Dependenzgrammatik des Lateinischen*. Vandenhoek & Ruprecht, 1975.

(Harris, 1979) Zellig Harris. *Mathematical Structures of Language*. Robert E. Krieger Publishing Company, Huntingtion, New York, 1979.

(Harrison, 1978) Michael A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, Reading (Mass.), 1978.

(Kayne, 1983) Richard Kayne. *Connectedness and Binary Branching*. Foris, Dordrecht, 1983.

(Kayne, 1994) Richard S. Kayne. *The Antisymmetry of Syntax*. Number 25 in Linguistic Inquiry Monographs. MIT Press, 1994.

(Keenan and Faltz, 1985) Edward L. Keenan and Leonard M. Faltz. *Boolean Semantics for Natural Language*. Number 23 in Synthese Language Library. Reidel, Dordrecht, 1985.

(Koster and Reuland, 1991) Jan Koster and Eric Reuland, editors. *Long-Distance Anaphora*. Cambridge University Press, Cambridge, 1991.

(Koster, 1986) Jan Koster. *Domains and Dynasties: the Radical Autonomy of Syntax*. Foris, Dordrecht, 1986.

(Kracht, 1993) Marcus Kracht. Splittings and the finite model property. *Journal of Symbolic Logic*, 58:139 – 157, 1993.

(Kracht, 1994) Marcus Kracht. Syntactic Codes and Grammar Refinement. *Journal of Logic, Language and Information*, 1994.

(Kracht, 1995) Marcus Kracht. Highway to the danger zone. *Journal of Logic and Computation*, 5:93 – 109, 1995.

(Laka, 1991) Itziar Laka. Negation in syntax: on the nature of functional categories and projections. *International Journal of Basque Linguistics and Philology*, 25:65 – 138, 1991.

(Lamb, 1966) Sydney M. Lamb. *Outline of Stratificational Grammar*. Washington, 1966.

(Larson, 1988) R. Larson. On the double object construction. *Linguistic Inquiry*, 19:335 – 391, 1988.

(Lasnik, 1976) Howard Lasnik. Remarks on coreference. *Linguistic Analysis*, 2:1–22, 1976.

(Levelt, 1991) Willem P. Levelt. *Speaking. From Intention to Articulation*. MIT Press, Cambridge (Mass.), 2 edition, 1991.

(Manzini, 1992) Maria R. Manzini. *Locality – A Theory and Some of Its Empirical Consequences*. Number 19 in Linguistic Inquiry Monographs. MIT Press, 1992.

(Morrill, 1994) Glyn V. Morrill. *Type Logical Grammar. Categorial Logic of Signs*. Kluwer Academic Publishers, Dordrecht, 1994.

(Parsons, 1994) Terence Parsons. *Events in the Semantics of English. A Study in Subatomic Semantics*. Number 19 in Current Studies in Linguistics. MIT Press, 1994.

(Pesetzky, 1995) David Pesetzky. *Zero Syntax. Experiencers and Cascades*. Number 27 in Current Studies in Linguistics. MIT Press, 1995.

(Rizzi, 1990) Luigi Rizzi. *Relativized Minimality*. MIT Press, Boston (Mass.), 1990.

(Roman, 1990) André Roman. *Grammaire de l'Arabe*. Number 1275 in Que Sais-Je? Presses Universitaires de France, 1990.

(Sells, 1985) Peter Sells. *Lectures on Contemporary Syntactic Theories*. Number 3 in CSLI Lecture Notes. CSLI, 1985.

(Stabler, 1992) Edward P. Stabler. *The Logical Approach to Syntax. Foundation, Specification and Implementation of Theories of Government and Binding*. ACL-MIT Press Series in Natural Language Processing. MIT Press, Cambridge (Mass.), 1992.

(Steedman, 1990) Mark Steedman. Gapping as Constituent Coordination. *Linguistics and Philosophy*, 13:207 – 263, 1990.

(Sternefeld, 1991) Wolfgang Sternefeld. *Syntaktische Grenzen. Chomskys Barrierentheorie und ihre Weiterentwicklungen*. Westdeutscher Verlag, Opladen, 1991.

(van Riemsdijk and Williams, 1986) H. van Riemsdijk and E. Williams. *Introduction to the Theory of Grammar*. MIT Press, Cambridge (Mass.), 1986.

(Watanabe, 1993) Akira Watanabe. *AGR-based case theory and its interaction with the A-bar system*. PhD thesis, MIT, 1993.

(Williams, 1994) Edwin Williams. *Thematic Structure in Syntax*. Number 23 in Linguistic Inquiry Monographs. MIT Press, 1994.