

# Lectures on Interpreted Languages and Compositionality

Marcus Kracht  
*Department of Linguistics, UCLA*  
*3125 Campbell Hall*  
*PO Box 951543*  
*Los Angeles, CA 90095-1543*  
[kracht@humnet.ucla.edu](mailto:kracht@humnet.ucla.edu)

and

*Magyar Tudományos Akadémia*  
*Nyelvtudományi Intézet*  
*Benczur u. 33*  
*H-1068 Budapest*  
[kracht@nytud.hu](mailto:kracht@nytud.hu)

October 10, 2006

## Introduction

This manuscript presents an outline of something which I like to call *metatheory of linguistics*. As the name says, it studies the properties of linguistic theories. More exactly, what I propose to study here is certain theoretical principles and their relationship with linguistic theories. This may be seen simply as theoretical linguistics but I'd like to propose it as a separate enterprise since it does not aim at studying the properties of grammars or frameworks as such. Rather, the aim is to find out in what way our initial assumption about the structure of language or linguistic theory can actually yield an insight into languages and what this insight consists in. We shall isolate a few metatheoretical principles and investigate their empirical potential in this way. One such principle is the *Principle of Compositionality*; another is *Leibniz' Principle*. It will emerge, for example, that the Principle of Compositionality has no empirical impact whatsoever unless we fix the input to be signs consisting of form and meaning; additionally, when defining form and meaning for natural languages we must make sure that there are restrictions on syntactic and semantic representations and functions. If this is guaranteed, however, we shall show there can be concrete results about the structure of natural languages.

Although in principle formulated in a neutral style, the motivation for this research is the belief that ultimately both the Principle of Compositionality and Leibniz' Principle are correct. Such beliefs are impossible to verify, of course, if, as is done here, they are taken as primitive assumptions from which certain conclusions are derived. On the other hand, it is my personal conviction that both principles are in principle falsifiable, and thus of an empirical nature. Abstractly it is possible to show that there are language for which they do not hold. Concretely, however, it is difficult to show that a given natural language fails them. This has to do with certain problems that are inherent in empirical research. One is that the results are never objective; they are obtained within a pretheoretical framework and therefore have to be used with care. Another is that the results are hardly ever categorical; they do not really yield a definite 'no' to the initial hypothesis. All they do is point in a certain direction.

A big problem area in this respect is the nature of semantics. It is, I guess,

clear that the study of languages as mere syntactic objects is a useless endeavour. Ultimately, the aim of linguistic theory must be to study syntactic objects insofar as they are either themselves meaningful expressions or parts thereof. However, large parts of linguistic theory refuse to deal with semantics in this way. I have sometimes met a certain unwillingness to acknowledge semantic data on the basis that it is unclear what this data consists in. For example, we may have doubts that two given expressions are synonymous (and whether there at all exist two synonymous expressions in a given language). But that in itself is an empirical question and has no bearing on the usefulness of semantics as such. Additionally, the same questions beset syntax as well. Grammaticality judgements are known to be elusive. Yet syntacticians use them without the slightest hesitation. If they would be similarly generous when it comes to semantics there would be no need to argue my case.

The Principle of Compositionality can be seen as an abstract requirement. The rationale for adopting it, however, comes from an assumption on the architecture of language that is not universally shared. In fact, as it turns out, there is hardly any linguistic theory that fully complies with this principle. This may be surprising since on the other hand linguists often judge theories on the basis of whether they are compositional or not. Thus, if we value compositionality so highly we ought to know what exactly makes a theory compositional. This is part of what these notes are about. Part of my claims may be contentious. For example, I claim below that indices are not part of a syntactic representation. This militates against a number of well-established theories, among them generative grammar and Montague grammar (!). It may therefore be thought that this diminishes the usefulness of the present approach. On the other hand, it is not my task to agree with a theory simply because it is popular. What is at stake is rather the very foundation on which the current theories are built. And in this respect it seems to me that linguistic theory on the whole suffers from a lack of understanding of how solid the ground is on which our theories are built. The notion of syntactic structure, for example, has become highly theory internal in generative grammar. This means that to argue that generative grammar has found out that sentences have such and such structure would simply be nonsense. The sentence structure depends on so many factors, each of which of highly abstract status (for example Kayne's principle of antisymmetry) that sentence structure is no longer a primitive notion by which we can compare generative grammar with other theories. Although one may dismiss sentence structure anyhow as epitheoretic, it is important to understand how much weight results on particular sentence structure actually have.

The current text is a development of ideas that I have first exposed in [Kracht, 2003]. Since then I have spent considerably energy in getting a clearer idea on the central notion of this book, namely compositionality. Additionally, I had the benefit of extended discussions with Kit Fine, Hans-Martin Gärtner, Fritz Hamm, Ben Keil, Ed Keenan, Udo Klein, Greg Kobele, Uwe Mönnich and Ed Stabler. Special thanks also to István Kenesei for his support. All of them have influenced my views on the subject in numerous ways. The responsibility for any occurring errors in this text remains entirely with me.

# Contents

<b>1</b>	<b>String Languages</b>	<b>7</b>
1.1	Languages and Grammars . . . . .	7
1.2	String Categories . . . . .	11
1.3	Syntactic Structure . . . . .	16
1.4	The Principle of Preservation . . . . .	22
<b>2</b>	<b>Compositionality and Leibniz' Principle</b>	<b>27</b>
2.1	Interpreted Languages and Grammars . . . . .	27
2.2	Do We Need Categories? . . . . .	32
2.3	Compositionality and Independence . . . . .	36
2.4	Leibniz' Principle . . . . .	44
<b>3</b>	<b>Meanings</b>	<b>49</b>
3.1	Desyntactified Meanings . . . . .	49
3.2	The Space of Meanings . . . . .	51
3.3	Linking Aspects . . . . .	55
<b>4</b>	<b>Examples</b>	<b>59</b>

4.1	Predicate Logic . . . . .	59
4.2	Concept Based Predicate Logic . . . . .	62
4.3	A Fragment of English . . . . .	67
4.4	Why Use Concepts? . . . . .	73

# Chapter 1

## String Languages

THIS chapter introduces the notion of a grammar. We shall describe how context free grammars and adjunction grammars fit the format described here. Then we shall study syntactic categories as they arise implicitly in the formulation of the grammar and then turn to the relationship between languages, grammars and surface tests to establish structure. We shall meet our first principle: the *principle of preservation*.

### 1.1 Languages and Grammars

Given a set  $A$ , we denote the set of strings over  $A$  by  $A^*$ . We write  $\vec{x}, \vec{y}$  (with an arrow) for arbitrary strings. Concatenation is either denoted by  $\vec{x}\vec{y}$  or by juxtaposition. Standard formal language defines languages as follows.

**Definition 1.1** *Let  $A$  be a finite set, the so-called **alphabet**. A **language over  $A$**  is a subset of  $A^*$ , the set of strings over  $A$ .*

A language is a set. A grammar on the other hand is a description of the language. There are two types of grammars: descriptive and generative. Descriptive grammars describe the strings of the language, while generative grammars describe a process that generates them. We shall delay a definition of descriptive grammars.

**Definition 1.2** Let  $F$  a set. A **signature** is a function  $\Omega$  from  $F$  to the set  $\mathbb{N}$  of natural numbers. Given  $f$ ,  $\Omega(f)$  is called the **arity** of  $f$ .  $f$  is a **constant** if  $\Omega(f) = 0$ .

If  $f$  has arity 2, for example, this means that it takes two arguments and yields a value. If  $f$  is a function on the set  $S$ , then  $f : S \times S \rightarrow S$ . We also write  $f : S^2 \rightarrow S$ . The result of applying  $f$  to the arguments  $x$  and  $y$  in that order is denoted by  $f(x, y)$ . If  $f$  is partial then  $f(x, y)$  need not exist. In this case we write  $f : S^2 \hookrightarrow S$ . We mention a special case, namely  $\Omega(f) = 0$ . By convention,  $f : S^0 \rightarrow S$ . Now,  $S^0 = \{\emptyset\}$ , and so  $f$  yields a single value if applied to  $\emptyset$ . However,  $\emptyset$  is simply the empty tuple in this connection, and we would have to write  $f()$  for the value of  $f$ . However, we shall normally write  $f$  in place of  $f()$ , treating  $f$  as if it was its own value. The 0-ary functions play a special role in this connection, since they shall form the *lexicon*.

**Definition 1.3** A **grammar over  $A$**  is a pair  $\langle \Omega, \mathcal{J} \rangle$ , where  $\Omega$  is a signature and for every  $f \in F$ ,  $\mathcal{J}(f) : (A^*)^{\Omega(f)} \hookrightarrow A^*$ .  $F$  is the set of **modes** of the grammar. The set  $\{f : \Omega(f) = 0\}$  is called the **lexicon** of  $G$ , and the set  $\{f : \Omega(f) > 0\}$  the set of **rules**.

Evidently, any  $f \in F$  (that is, every mode) is either in the lexicon or in the set of rules. Notice that there are no requirements on the functions, not even that they be computable. We shall introduce restrictions on the functions as we go along. The lexicon is not always considered part of the grammar. We make no principled decision here; it is just easier not to have to worry about rules and lexicon separately.

**Example 1.** This is one of our main examples: it will be called *the language of equations*.  $A = \{\emptyset, 1, +, -, (, ), =\}$ .  $F = \{f_0, f_1, f_2, f_3, f_4, f_5, f_6\}$ .  $\Omega(f_0) = \Omega(f_1) = 0$ ,  $\Omega(f_2) = \Omega(f_3) = 1$ ,  $\Omega(f_4) = \Omega(f_5) = \Omega(f_6) = 2$ .  $\vec{x}$  is **binary** if it only contains  $\emptyset$

and 1;  $\vec{x}$  is a **term** if it does not contain =.

$$\begin{aligned}
 \mathcal{J}(f_0)() &:= \mathbf{0} \\
 \mathcal{J}(f_1)() &:= 1 \\
 \mathcal{J}(f_2)(\vec{x}) &:= \begin{cases} \vec{x}^\wedge \mathbf{0} & \text{if } \vec{x} \text{ is binary} \\ \text{undefined} & \text{else} \end{cases} \\
 \mathcal{J}(f_3)(\vec{x}) &:= \begin{cases} \vec{x}^\wedge 1 & \text{if } \vec{x} \text{ is binary} \\ \text{undefined} & \text{else} \end{cases} \\
 \mathcal{J}(f_4)(\vec{x}, \vec{y}) &:= \begin{cases} (\vec{x}^\wedge + \vec{y}^\wedge) & \text{if } \vec{x}, \vec{y} \text{ are terms} \\ \text{undefined} & \text{else} \end{cases} \\
 \mathcal{J}(f_5)(\vec{x}, \vec{y}) &:= \begin{cases} (\vec{x}^\wedge - \vec{y}^\wedge) & \text{if } \vec{x}, \vec{y} \text{ are terms} \\ \text{undefined} & \text{else} \end{cases} \\
 \mathcal{J}(f_6)(\vec{x}, \vec{y}) &:= \begin{cases} \vec{x}^\wedge = \vec{y}^\wedge & \text{if } \vec{x}, \vec{y} \text{ are terms} \\ \text{undefined} & \text{else} \end{cases}
 \end{aligned}
 \tag{1.1}$$

The strings that this grammar generates are of the following form. They are either strings of 0s and 1s, for example  $\mathbf{010}$ ,  $111\mathbf{0}1$ , or they are terms, like  $(1+(\mathbf{0}1-\mathbf{10}1))$ ; or they are equations between two such terms, like  $(1+\mathbf{10})=11$ . (A single numeral expression is also a term.) ♣

**Definition 1.4** Let  $V = \{x_i : i \in \mathbb{N}\}$  a set of variables. Let  $\Omega$  be a signature. An  $\Omega$ -**term** is a sequence of the following form.

- ❶  $x_i, i \in \mathbb{N}$ ,
- ❷  $f$ , where  $\Omega(f) = 0$ ,
- ❸  $ft_0 \cdots t_{n-1}$ , where  $n = \Omega(f)$  and for every  $i < n$   $t_i$  is an  $\Omega$ -term.

If  $X \subseteq V$ , the symbol  $\text{Tm}_\Omega(X)$  denotes the set of all  $\Omega$ -terms which can be built over the set  $X$ . The set  $\text{Tm}_\Omega(\emptyset)$  is of special importance. It is the set of **constant  $\Omega$ -terms**.

Notice that the second case is a subcase of the third; it is listed separately for better understanding (but I have nowhere stated that  $n$  must be greater than 0).

Given a grammar  $G$  we can define the interpretation  $\iota_G(t)$  of a constant term  $t$ .

- ❶  $\iota_G(f) := \mathcal{J}(f)$  if  $\Omega(f) = 0$ ,
- ❷  $\iota_G(ft_0 \cdots t_{n-1}) := \mathcal{J}(f)(\iota_G(t_0), \dots, \iota_G(t_{n-1}))$ , where  $n = \Omega(f)$ .

If the grammar is clear from the context, we shall write  $\iota(t)$  in place of  $\iota_G(t)$ . Continuing our example, we have

$$\begin{aligned}
 (1.2) \quad \iota(f_4 f_3 f_0 f_2 f_1) &= (\iota(f_3 f_0) + \iota(f_2 f_1)) \\
 &= (\iota(f_0)1 + \iota(f_2 f_1)) \\
 &= (\iota(f_0)1 + \iota(f_1)\mathbf{0}) \\
 &= (\mathbf{0}1 + \iota(f_1)\mathbf{0}) \\
 &= (\mathbf{0}1 + 1\mathbf{0})
 \end{aligned}$$

This establishes the interpretation of constant terms. For terms containing variables the interpretation must be a function from values to these variables to strings. Here is a way to implement this idea. The interpretation of a term is a partial function from  $(A^*)^{\mathbb{N}}$  to  $A^*$ . Here, an infinite sequence  $\bar{s} := \langle s_0, s_1, \dots \rangle$  is meant to be an assignment of strings to the variables. Namely, the string  $s_i$  is supposed to be the value of the variable  $x_i$ . To implement this put

- ❶  $\iota'(x_i)(\bar{s}) := s_i$ ,
- ❷  $\iota'(f)(\bar{s}) := \mathcal{J}(f)$  if  $\Omega(f) = 0$ ,
- ❸  $\iota'(ft_0 \cdots t_{n-1})(\bar{s}) := \mathcal{J}(f)(\iota'(t_0)(\bar{s}), \dots, \iota'(t_{n-1})(\bar{s}))$ , where  $n = \Omega(f)$ .

This latter definition will not often be used, but is put here for future reference. Notice that if the strings functions are partial, some of the  $\iota(t)$  may also be partial functions.

Since the string functions may be partial not every constant term has a value. Thus,  $\iota(t)$  may be undefined. We call

$$(1.3) \quad \text{dom}(\iota) := \{t \in \text{Tm}_{\Omega}(\emptyset) : \iota(t) \text{ is defined}\}$$

the set of **orthographically definite terms**. The term  $f_4 f_3 f_0 f_2 f_1$  is orthographically definite, while the term  $f_4 f_0 f_4$  is not. This is because once  $f_4$  has been used,

it introduces the symbol  $=$ , and none of the modes can apply further. Notice that for a grammar  $G$ , the language can simply be defined as

$$(1.4) \quad L(G) := \{\iota(t) : t \in \text{Tm}_\Omega(\emptyset)\}$$

Given a grammar  $G$  and a string  $\vec{x}$ , we call a term  $t$  an **analysis** of  $\vec{x}$  if  $\iota(t) = \vec{x}$ . A string may have several analysis terms. In this case we say that it is **ambiguous**. If it has none it is called **ungrammatical**.

**Exercise 1.** Describe the set of orthographically definite structure terms for the language of equations.

## 1.2 String Categories

The string functions discussed in the previous section are partial. This means effectively that there are different types of strings. Each grammar therefore defines a finite set of string categories in the following way.

**Definition 1.5** *Let  $G$  be a grammar. We write  $\vec{x} \sim_G \vec{y}$  if for every term  $t(x_0)$ ,  $\iota'(t)(\vec{x})$  is defined iff  $\iota'(t)(\vec{y})$  is defined. We write  $[\vec{x}]_G := \{\vec{y} : \vec{x} \sim_G \vec{y}\}$ . These sets are called the **syntactic categories** of  $G$ .*

Notice that the set of strings on which *no* function is defined also is a syntactic category. There are two reasons why we need them. One is that the definitions become simpler. Another is that in certain cases some members of the language  $L(G)$  may even be of that kind. The example of the previous section shows that  $L(G)$  may be contained in this set.

There need not be finitely many equivalence classes as the following example shows.  $A := \{a\}$ .  $\Omega(f) = 1$ .

$$(1.5) \quad f(a^n) := \begin{cases} a^{n-1} & \text{if } n \geq 0 \\ \text{undefined} & \text{else} \end{cases}$$

In this case we have that if  $m > n$ :  $f^n(a^m) = a^{m-n}$  and  $f^m(a^m) = \varepsilon$ . However, for  $n > m$ ,  $f^n(a^m)$  is undefined. Thus, for this grammar  $G$ ,  $a^m \sim_G a^n$  iff  $m = n$ , and so there are infinitely many equivalence classes.

We shall now turn to an important class of grammars, namely *context free grammars*. Recall that these grammars are formulated using an additional set of symbols, called **nonterminals**. Let  $N$  be the set of nonterminals. Then a CFG generates strings from  $(A \cup N)^*$ . Also, the standard way to look at CFGs is top down: it starts with a string consisting of a single symbol, the so called start symbol. The grammar of the binary equations can be given as follows.

**Example 2.** We continue the language of equations (Example 1 on Page 8). The grammar  $G_Q$  consists of the following alphabet of terminals  $A := \{0, 1, +, -, (, ), =\}$ , the alphabet of nonterminals  $N := \{E, B, T\}$ , start symbol  $E$  and the following set  $R$  of rules:

$$(1.6) \quad \begin{aligned} E &\rightarrow T=T \\ T &\rightarrow (T+T) \mid (T-T) \mid B \\ B &\rightarrow B0 \mid B1 \mid 0 \mid 1 \end{aligned}$$

Each derivation starts with  $E$ . Thus

$$(1.7) \quad C = \langle A, N, E, R \rangle$$

Recall that ‘|’ is an abbreviation. It allows to group together rules with the same left hand side. Here is an example of a derivation:

$$(1.8) \quad \begin{aligned} &E \\ &(T+T) \\ &(B+T) \\ &(0+T) \\ &(0+B) \\ &(0+B1) \\ &(0+11) \end{aligned}$$

In each step we replace a single occurrence of a nonterminal by a corresponding right hand side. ♣

As matters stand, CFGs are not grammars in the sense defined in the previous section. Two things need to be changed; the first is that we must reverse the direction of the rules. We need to think of the rules as going from right to left. The reason for this becomes clear when we apply the second step: the elimination

of nonterminal categories. For notice that the additional symbols are not part of the original language. They have been added. However, turning back to the original grammar we can see that the auxiliary symbols have been added to code the syntactic categories that the grammar defines.

**Definition 1.6** *Let  $G$  be a context free grammar. We write  $\vec{x} \Rightarrow_G \vec{y}$  if there are strings  $\vec{u}, \vec{v}$  and  $\vec{w}$  and a nonterminal  $X$  such that  $\vec{x} = \vec{u}X\vec{w}$ ,  $\vec{y} = \vec{u}\vec{v}\vec{w}$  and  $X \rightarrow \vec{v}$  is a rule of  $G$ . We write  $\vec{x} \Rightarrow_G^* \vec{y}$  if there is an  $n$  and strings  $\vec{z}_i$ ,  $i < n$ , such that*

$$(1.9) \quad \vec{x} = \vec{z}_0 \Rightarrow_G \vec{z}_1 \Rightarrow_G \vec{z}_2 \cdots \Rightarrow_G \vec{z}_{n-1} = \vec{y}$$

*If  $n = 0$  then  $\vec{x} = \vec{y}$ , which is admitted.*

Call  $\vec{x}$  an  **$X$ -string** if  $X \Rightarrow_G^* \vec{x}$ . Write  $L_X(G)$  for the set of  $X$ -strings of  $G$ . In our example  $L_E(G_Q)$  is the set of equations; these are strings of the form  $\vec{x}=\vec{y}$ , where both  $\vec{x}$  and  $\vec{y}$  are T-strings. T-strings are terms; these are strings of the form (a)  $\vec{x}$ , where  $\vec{x}$  consists of  $\mathbf{0}$  and  $\mathbf{1}$  only (a number expression, or a B-string), (b)  $(\vec{x}+\vec{y})$  where  $\vec{x}$  and  $\vec{y}$  are T-strings, or (c)  $(\vec{x}-\vec{y})$  where  $\vec{x}$  and  $\vec{y}$  are T-strings. Finally, the B-strings are exactly the strings from  $\{\mathbf{0}, \mathbf{1}\}^+$ .

We can now describe how to reverse this context free grammar. For each rule  $\rho$  we introduce a function symbol  $f_\rho$ . The arity of  $f_\rho$  equals the number of nonterminals on the right hand side of the rule. For example,  $\rho = \mathbf{B} \rightarrow \mathbf{B}\mathbf{1}$  is a rule of  $G_Q$ , and so we have a symbol  $f_\rho$  with  $\Omega(f_\rho) = 1$ . The function takes a B-string  $\vec{x}$  and appends  $\mathbf{1}$ . Hence:

$$(1.10) \quad f_\rho(\vec{x}) := \begin{cases} \vec{x}\mathbf{1} & \text{if } \vec{x} \text{ is a B-string} \\ \text{undefined} & \text{else} \end{cases}$$

Similarly, if  $\rho' = \mathbf{T} \rightarrow (\mathbf{T}+\mathbf{T})$  we postulate a symbol  $f_{\rho'}$  with  $\Omega(f_{\rho'}) = 2$  and which acts as follows:

$$(1.11) \quad f_{\rho'}(\vec{x}, \vec{y}) := \begin{cases} (\vec{x}+\vec{y}) & \text{if } \vec{x} \text{ and } \vec{y} \text{ are T-strings} \\ \text{undefined} & \text{else} \end{cases}$$

As we have briefly noted above, the properties ‘B-string’, ‘T-string’ and so on can actually be defined without taking recourse to the grammar.

In general, the reversal of rules is as follows. Let

$$(1.12) \quad A \rightarrow \vec{x}_0 \vec{B}_0 \vec{x}_1 B_1 \cdots B_{n-2} \vec{x}_{n-1}$$

be a rule. Then we add a symbol  $f_\rho$  of arity  $n - 1$  which acts as follows:

$$(1.13) \quad f_\rho(\vec{y}_0, \dots, \vec{y}_{n-2}) := \begin{cases} \vec{x}_0 \vec{y}_0 \vec{x}_1 \vec{y}_1 \cdots \vec{y}_{n-2} \vec{x}_{n-1} & \text{if for all } i < n - 1: \\ & \vec{y}_i \text{ is a } B_i\text{-string} \\ \text{undefined} & \text{else} \end{cases}$$

This is a grammar in the sense of the previous section. The strings it generates are all  $X$ -strings, where  $X$  is a nonterminal of the CFG we started off with. This is an important fact: there is no direct way to generate only the language  $L(G)$ , which are the  $S$ -strings for the start symbol  $S$ . There are grammars that are designed to do exactly this. These are the string adjunction grammars, which are defined as follows.

**Definition 1.7** A *2-context* is a triple  $\gamma = \langle \vec{u}, \vec{v}, \vec{w} \rangle$ . The result of inserting a pair  $\langle \vec{x}, \vec{y} \rangle$  into  $\gamma$  is defined as follows:

$$(1.14) \quad \gamma(\langle \vec{x}, \vec{y} \rangle) := \vec{u} \vec{x} \vec{v} \vec{y} \vec{w}$$

A *2-locale* is a set of 2-contexts. A *string adjunction rule* is a pair  $\langle \langle \vec{x}, \vec{y} \rangle, \Lambda \rangle$ , where  $\Lambda$  is a locale.

**Definition 1.8** A *string adjunction grammar* is a pair  $A = \langle S, R \rangle$ , where  $S$  is a finite set of strings and  $R$  a finite set of string adjunction rules. We say that  $A$  *generates*  $\vec{y}$  in  $n$ -steps if the following holds:  $n = 0$  and  $\vec{y} \in S$  or  $n > 0$  and there is a  $\vec{z}$  such that  $A$  generates  $\vec{z}$  in  $n - 1$  steps and there is a rule  $\langle \langle \vec{x}_0, \vec{x}_1 \rangle, \Lambda \rangle$  and  $\gamma = \langle \vec{u}, \vec{v}, \vec{w} \rangle$  such that  $\vec{z}$  possesses the decomposition  $\vec{z} = \gamma(\langle \varepsilon, \varepsilon \rangle) = \vec{u} \vec{v} \vec{w}$  and

$$(1.15) \quad \vec{y} = \gamma(\langle \vec{x}_0, \vec{x}_1 \rangle) = \vec{u} \vec{x}_0 \vec{v} \vec{x}_1 \vec{w}$$

$L(A)$  denotes the set of strings that can be generated in a finite number of steps.

**Example 3.** We shall now give a presentation of the E-strings of the grammar from Example 2 using a string adjunction grammar. We use

$$(1.16) \quad S = \{0=0, 0=1, 1=0, 1=1\}$$

The rules are as follows. Let  $\Lambda_1$  be the set of triples  $\langle \vec{u}x, \vec{v}, \vec{w} \rangle$  such that  $c$  is either 0 or 1 and  $\vec{v} \vec{w}$  does not begin with 0 or 1. Let  $\Lambda_2$  be the set of triples of the form

$\langle \vec{u}, \vec{v}, \vec{w} \rangle$ , where both  $\vec{u}$  does not and  $\vec{w}$  does not begin with  $\mathbf{0}$  or  $1$ , while  $\vec{v} \in \{\mathbf{0}, 1\}^*$ .

$$(1.17) \quad R = \{ \langle \langle \mathbf{0}, \varepsilon \rangle, \Lambda_1 \rangle, \\ \langle \langle 1, \varepsilon \rangle, \Lambda_1 \rangle, \\ \langle \langle \varepsilon, \mathbf{0} \rangle, \Lambda_1 \rangle, \\ \langle \langle \varepsilon, 1 \rangle, \Lambda_1 \rangle, \\ \langle \langle (, +\mathbf{0}) \rangle, \Lambda_2 \rangle, \\ \langle \langle (, +1) \rangle, \Lambda_2 \rangle \}$$

Here is an example of a derivation:

$$(1.18) \quad \begin{aligned} &\mathbf{0}=1 \\ &(\mathbf{0}+1)=1 \\ &(\mathbf{0}+1\mathbf{0})=1 \\ &(\mathbf{0}+(1\mathbf{0}+1))=1 \end{aligned}$$

The first line is in  $S$ . To get from the first line to the second we choose a decomposition  $\mathbf{0}=1 = \varepsilon \wedge \mathbf{0} \wedge =1$ . Thus, choose  $\gamma = \langle \varepsilon, \mathbf{0}, =1 \rangle$ . This is in  $\Lambda_2$ , since  $\varepsilon$  does not end in  $\mathbf{0}$  or  $1$ , and  $=1$  does not begin with  $\mathbf{0}$  or  $1$ . Thus we can apply the rule  $\langle \langle (, +1) \rangle, \Lambda_2 \rangle$ . We get

$$(1.19) \quad \gamma(\langle \langle (, 1) \rangle \rangle) = \varepsilon \wedge (\wedge \mathbf{0} \wedge +1) \wedge =1 = (\mathbf{0}+1)=1$$

♣

The question that arises is whether adjunction grammars can be brought into the format required in Section 1.1. For the strings from  $S$  this is unproblematic. For each string  $\vec{x} \in S$  we introduce a 0-ary rule  $f_{\vec{x}}$  with value  $\vec{x}$ . The adjunction rules however do pose a problem. Look at the rule

$$(1.20) \quad \rho = \langle \langle (, +\mathbf{0}) \rangle, \Lambda_2 \rangle$$

For each  $n$  there is a string  $\vec{u}_n$  in the language such that  $\vec{u}_n$  possesses at least  $n$  different decompositions into three strings that form a 2-locale from  $\Lambda_2$ ; and such that adjoining the pair  $\langle (, +\mathbf{0}) \rangle$  will produce a different result. Namely, define

$$(1.21) \quad \begin{aligned} \vec{u}_2 &:= 1=1 \\ \vec{u}_3 &:= (1+1)=1 \\ \vec{u}_4 &:= (1+(1+1))=1 \\ \vec{u}_5 &:= (1+(1+(1+1)))=1 \\ &\dots \end{aligned}$$

There are two solutions to this problem. One is to artificially restrict the adjunction sites. Another is to allow for the functions to be *indeterminate*, that is, to have several outputs. The second solution becomes problematic when we turn to interpreted languages, for the indeterminacy must be accompanied by semantic indeterminacy, which is not always the case—if not rarely. The first solution likewise seems problematic (although I have not checked the exact details). It is not clear to me how to effectively block adjunction at all but a bounded number of places. In tree adjoining grammars it is possible to explicitly mark positions for the trees that can be adjoined. String adjoining grammars do not possess such a device, but it can certainly be added.

The choice between string adjunction grammars and CFGs is therefore mainly determined by the question in which way we want to think of our strings. If we think of them as being divided into several types of strings, and furthermore, if we think that complex strings are made of less complex strings of possibly different category, then we opt for a CFG. If we think that complex strings are made from simpler strings, but that the simpler strings are essentially of the same category, then we opt for string adjunction grammars.

### 1.3 Syntactic Structure

Contemporary linguistics is built on the assumption that what matters is not the string but rather the *structure*. Structure usually means tree structure. It has been stressed by Chomsky that rules operate on constituents and not on strings. Formally, however, it is not clear whether the structure is necessarily represented. To see this, let us take a look at CFGs. Given a string  $\vec{x}$  and a grammar  $G$  that generates it,  $G$  assigns a structure to  $\vec{x}$  through a derivation in the following way.

We shall construct a derivation of the string  $\vec{x}$ . We start with the tree consisting of a single node labelled  $S$ , where  $S$  is the start symbol. We put  $\vec{x}_0 := S$ . At each step we construct a string  $\vec{x}_i$  and a tree  $\mathfrak{T}_i$  whose yield is  $\vec{x}_i$ . When we are done, we have  $\vec{x}_n = \vec{x}$ .  $\vec{x}_i$  is a string consisting of both terminal and nonterminal symbols. To ease the definitions we disallow terminal nodes which are empty. If we apply a rule  $X \rightarrow \vec{Y}$  to the  $m$ th symbol from the left in the string  $\vec{x}_i$ , we get  $\mathfrak{T}_{i+1}$  from  $\mathfrak{T}_i$  by replacing the  $m$ th leaf of  $\mathfrak{T}_i$  by the local tree defined by the rule  $X \rightarrow \vec{Y}$ .

We can code the derivation into the string by assuming in place of  $G$  the brack-

eted grammar  $G^b$ . This grammar is define as follows. We introduce for each nonterminal symbol  $X$  a pair of brackets  $(_X$  and  $)_X$ . Let  $\rho = X \rightarrow \vec{Y}$  be a rule. Then

$$(1.22) \quad \rho^b := X \rightarrow ({}_X \vec{Y})_X$$

$G^b$  contains in place of the rules  $R$  the set

$$(1.23) \quad R^b := \{\rho^b : \rho \in R\}$$

Let  $\vec{x}$  be a string. Each derivation  $D$  of  $\vec{x}$  can be paralleled by a derivation  $D^b$ , which uses  $\rho^b$  in place of  $\rho$ , for each rule  $\rho \in R$ , at the appropriate places. Thus mapping the original derivation into a bracketed derivation  $D^b$  we find the string  $\vec{x}_b$ , which contains a record of the derivation.  $\vec{x}$  is found by deleting the brackets and the category symbols. More exactly, define a homomorphism  $d$  as follows.

$$(1.24) \quad d(a) := \begin{cases} \varepsilon & \text{if for some } X: a = ({}_X \text{ or } a = )_X \\ a & \text{else} \end{cases}$$

$$d(x_0 x_1 \cdots x_{n-1}) := d(x_0) d(x_1) \cdots d(x_{n-1})$$

Notice that the mapping  $d$  is many to one, since a given string can have many derivations. Notice also that there may be derivations that lead to the same bracketed string. Thus the structure is intermediate between the string and the derivation, adding detail to the string but not enough to recover the entire derivation.

**Example 4.** Let  $G = \langle \{a, b\}, \{E, A, B\}, E, R \rangle$  where  $R$  contains the following rules:

$$(1.25) \quad \begin{aligned} E &\rightarrow AB \mid BA \mid EE \\ A &\rightarrow AE \mid EA \mid a \\ B &\rightarrow BE \mid EB \mid b \end{aligned}$$

Now  $G^b = \langle \{a, b\}, \{E, A, B, ({}_E, )_E, ({}_A, )_A, ({}_B, )_B\}, E, R^b \rangle$

$$(1.26) \quad \begin{aligned} E &\rightarrow ({}_E AB)_E \mid ({}_E BA)_E \mid ({}_E EE)_E \\ A &\rightarrow ({}_A AE)_A \mid ({}_A EA)_A \mid a \\ B &\rightarrow ({}_B BE)_B \mid ({}_B EB)_B \mid b \end{aligned}$$

Now the string  $abab$  can be derived in several ways. One is given by the sequence  $E, EE, ABE, ABAB$ , and so on; another is given by the sequence  $E, AB, AEB, ABAB$ , and so on. These derivations give rise to the following bracketed strings:

$$(1.27) \quad \begin{aligned} & (E(E(Aa)_A(Bb)_B)_E(E(Aa)_A(Bb)_B)_E)_E \\ & (E(Aa)_A(B(E(Bb)_B(Aa)_A)_E(Bb)_B)_B)_E \end{aligned}$$

You may check that erasing the brackets brings back the original string. The derivation  $E, EE, EAB, ABAB$  on the other hand yields the first string again. ♣

Notice however that the composition of the string  $\vec{x}$  in the inverted context free grammar requires neither the structure nor the derivation as a record. It can work all by itself. This is because the string alone suffices. This means in turn that grammatical structure does not have to be present in the representation; it arises only if we look at the derivational history of the string. If that is so, it is actually an *epiphenomenon*. It may be used in theoretical discourse but is in principle eliminable. This will have to be reassessed when we turn to interpreted grammars. We discuss the definitions and results first in the context of CFGs.

**Definition 1.9** Let  $\vec{x}$  be a string. An **occurrence** of  $\vec{y}$  in  $\vec{x}$  is a pair  $C = \langle \vec{u}, \vec{v} \rangle$  such that

$$(1.28) \quad C(\vec{y}) := \vec{u}\vec{y}\vec{v} = \vec{x}.$$

Given a grammar  $G$ , and a derivation  $D$  we can now assign constituent occurrences of substrings as follows. The definition is by induction on the length of the derivation and allows for derivations of strings containing both terminal and nonterminal symbols. Case 1.  $D$  has length 0.  $\vec{x} = S$ . The only constituent occurrence is  $O(D) := \{\langle \varepsilon, \varepsilon \rangle\}$ . Case 2.  $D$  is obtained from  $D'$  by applying  $X \rightarrow \vec{Y}$  to an occurrence  $C = \langle \vec{u}, \vec{v} \rangle$  of  $X$ . Then the new constituent occurrences are all occurrences of the following form. Let  $\vec{z}_+$  be the string obtained from  $\vec{z}$  by replacing a designated occurrence of  $X$  by  $\vec{Y}$ . Let  $\vec{Y} = Y_0Y_1 \cdots Y_n$ .  $O(D)$  is the union of four sets. (A) The set of all  $\langle \vec{u}Y_0 \cdots Y_{i-1}, Y_{i+1} \cdots Y_n\vec{v} \rangle$  where  $\langle \vec{u}, \vec{v} \rangle$  is the designated occurrence of  $X$ , (B) the set of all  $C$ , where the occurrence of  $X$  is not in  $\vec{u}$  or  $\vec{v}$ , (C) the set of all  $\langle \vec{u}_+, \vec{v} \rangle$  where  $\vec{u}$  contains the designated occurrence of  $X$ , (D) the set of all  $\langle \vec{u}, \vec{v}_+ \rangle$  where  $\vec{v}$  contains the designated occurrence of  $X$ . This definition basically repeats what is intuitively known. Moreover, from the derivation we can uniquely assign a category to the string occurrence. The following formalises the known substitution principle.

**Definition 1.10** Let  $G$  be a CFG,  $D$  a derivation of the string  $\vec{x}$  and  $C$  is an occurrence of  $\vec{y}$  in  $\vec{x}$ . If  $C \in O(D)$  then  $C$  is said to be a **constituent occurrence of  $\vec{y}$  in  $\vec{x}$** . If  $C \notin O(D)$ , the occurrence is said to be **accidental A-occurrence** if  $\vec{y} \in L_A(G)$ .

**Example 5.** We continue Example 4 above. The first derivation given by the sequence E, EE, ABE, ABAB, aBAB, abAB, abaB, abab. In the string we have the constituent occurrences  $\langle \varepsilon, \varepsilon \rangle$ ,  $\langle \varepsilon, ab \rangle$ ,  $\langle AB, \varepsilon \rangle$  of category E; the occurrences  $\langle \varepsilon, bab \rangle$  and  $\langle ab, b \rangle$  of category A; and the occurrences  $\langle a, ab \rangle$  and  $\langle aba, \varepsilon \rangle$  of category B. The string ab has an accidental occurrence  $\langle a, b \rangle$ . The string aa has no accidental occurrence although it is a substring of aabb. ♣

**Proposition 1.11** Let  $G$  be a CFG,  $\vec{x} \in L(G)$  and  $D$  a derivation. Fix a constituent occurrence of  $\vec{y}$  in  $\vec{x}$  under  $D$ . If  $\vec{y}$  occurs as  $A$  in the context  $C$ , and  $\vec{z}$  is any string of category  $A$  of  $G$ , then  $C(\vec{z}) \in L(G)$ .

Suppose now that we wish to give a syntactic analysis of a string language  $L$ . We assume that the analysis is given in terms of a CFG. If that is so, we know that the set of strings of  $L$  fall into finitely many classes, say,  $S_i$  for  $i < n$ , and that if  $\vec{x}, \vec{y} \in S_i$ , each constituent occurrence of  $\vec{x}$  can be substituted by  $\vec{y}$  and each constituent occurrence of  $\vec{y}$  can be substituted by  $\vec{x}$ . This superficially looks like a way to discover the grammar behind a given language.

The problem with this idea is that we do not know whether a given occurrence is a constituent occurrence. However there is one exception: a single letter wherever it occurs can only occur as a constituent on condition that the grammar contains no syncategorematic occurrences of symbols. It is easy to massage any CFG into such a form without losing anything.

**Example 6.** The language of equations. In the form presented in Example 2 on Page 12. This grammar introduces =, the operation symbols and the brackets in a syncategorematic way. It can be reformulated as follows. The original rule set is

$$\begin{aligned}
 & E \rightarrow T=T \\
 (1.29) \quad & T \rightarrow (T+T) \mid (T-T) \mid B \\
 & B \rightarrow B0 \mid B1 \mid 0 \mid 1
 \end{aligned}$$

Now introduce a nonterminal for each symbol. For example, introduce  $O$ ,  $C$ ,  $Q$  together with the unary rules

$$(1.30) \quad \begin{aligned} O &\rightarrow ( \\ C &\rightarrow ) \\ P &\rightarrow + \\ M &\rightarrow - \\ Q &\rightarrow = \end{aligned}$$

Next replace the occurrence of the syncategorematic symbols above by the corresponding nonterminal:

$$(1.31) \quad \begin{aligned} E &\rightarrow TQT \\ T &\rightarrow OTPTC \mid OTMTC \mid B \\ B &\rightarrow B\emptyset \mid B1 \mid \emptyset \mid 1 \end{aligned}$$

It is possible to simplify this grammar; we group  $P$  and  $M$  into just one symbol, say,  $H$ . Then we have the following rule set:

$$(1.32) \quad \begin{aligned} O &\rightarrow ( \\ C &\rightarrow ) \\ H &\rightarrow + \mid - \\ Q &\rightarrow = \\ E &\rightarrow TQT \\ T &\rightarrow OTHTC \mid B \\ B &\rightarrow B\emptyset \mid B1 \mid \emptyset \mid 1 \end{aligned}$$

♣

**Definition 1.12** A CFL is called *natural* if every substitution class contains a string of length 1, that is, consisting of a single letter.

The language defined above is not natural. This is because the set of E-strings (which form a substitution class!) consists of strings of length at least 3: an equation sign, and two terms on either side. Terms may not be empty, they have length at least 1.

Natural languages can easily be turned into CFGs. Just observe that for each letter  $a$  there is a substitution class  $[a]_L$ . Let  $N_a$  be the nonterminal representing this class (if  $[a]_G = [b]_G$  then also  $N_a = N_b$ .) The rules are of the form

$$(1.33) \quad \begin{array}{ll} N_a \rightarrow a & a \in A \\ N_a \rightarrow N_{c_0}N_{c_1} \cdots N_{c_n} & c_0c_1 \cdots c_n \in [a]_L \end{array}$$

This set is typically infinite, but a finite subset is enough to generate  $L$ , by assumption on  $L$ .

We should however note that the procedure can always be applied. We shall now turn now to the abstract case.

**Definition 1.13** *Let  $u$  and  $v$  constant  $\Omega$ -terms and  $G$  a grammar. We say that  $u$  and  $v$  are **categorially equivalent**, in symbols  $u \sim_G v$ , if for all terms  $s(x)$ :  $s(u)$  is defined iff  $s(v)$  is. They are **intersubstitutable**, in symbols  $\vec{u} \approx_G \vec{v}$ , iff they are categorially equivalent and  $s(u) \in L(G)$  iff  $s(v) \in L(G)$ .*

This definition does not talk about strings; it talks about terms. This is because the term may be very complex while the string is very simple. Moreover, in absence of any condition on the form of the rules it is not possible to assign any sensible structure to the string.

**Example 7.** Here is a context sensitive grammar, consisting of the following rules.

$$(1.34) \quad \begin{array}{l} S \rightarrow ATB \\ T \rightarrow x \mid xT \\ Ax \rightarrow xA \\ AB \rightarrow y \end{array}$$

In a derivation, first  $A$  is generated to the left of the string. However, when the last rule applies, it has to be to the right. The system of constituents formed by this grammar is quite confusing. It puts the occurrence of  $y$  into a constituent with all occurrences of  $x$  (for each occurrence of  $x$  there is a separate constituent, though).

♣

Notice also that adjunction grammars in the general form may fail to allow for an unequivocal assignment of structure. This is why tree adjunction grammars

work differently from string adjunction grammars. In TAGs the constituent structure is by definition preserved while in string adjunction grammars it need not be.

**Exercise 2.** Show that the substitution classes of a context free grammar (constituted as a grammar in the sense of this book in the straightforward way) are of the following form. Let  $N$  be the set of nonterminals, and  $P \subseteq N$ . Then a string  $\vec{x}$  is said to be of *class*  $P$  if for all  $Y \in N$ :  $Y \Rightarrow^* \vec{x}$  iff  $Y \in P$ .

## 1.4 The Principle of Preservation

We have seen in the previous section that substitution is rather difficult to handle. We propose here two principles that simplify the situation.

**Principle 1 (Structure)** *Exponents are sequences of strings.*

This is on the one restrictive, on the other hand more general than one expects at first sight. First of all, we have not said anything at all about the alphabet from which the strings are formed. In conjunction with the Principle of Structure Preservation this will simply be equivalent to saying that letters are alphabetic letters; but I think that matters are not that easy. The problems of this viewpoint will be discussed below. Let us for the moment remain with the idea that the alphabet is simply the standard typographical alphabet. Then exponents are strings of that alphabet—or, as it happens, sequences thereof. This latter qualification is important. Consider the following principle.

**Principle 2 (Structure Preservation)** *A rule may not break any string of the exponent or delete any parts of it.*

This means that if an exponent has the form  $\langle \vec{x}_0, \dots, \vec{x}_{n-1} \rangle$  then in a sentence of the language we must be able to find occurrences of  $\vec{x}_0, \vec{x}_1$  and so on which correspond to the strings in our exponent.

What these principle forbid is deletion of any kind, or breaking of constituents. However, what they *do* allow is discontinuity. A constituent may consist of a bounded number of parts. Typically, we find that constituents consist of just 1 or 2. An example of the latter kind are the verbs of German (after verb second has

applied), the crossed dependencies in Dutch infinitives, and split-DPs. Occasionally we find languages that seem to have arbitrarily fragmented DPs, like Warlpiri or Jiwari. However, in the case of these languages we might as well claim that they do not have such phrases at all. This is an empirical question.

We have so far only spoken about breaking or deleting strings.

**Principle 3 (Syncategorematicity Prohibition)** *A rule may not add any occurrence of a given symbol.*

This is somewhat harder to pin down exactly; the principle does allow for complete reduplication (as in Malay), and it also allows for partial reduplication, as long as the parts can be represented as strings. The way it does so is by stipulating that a given string may be repeated. This in fact does *not* mean that a fixed symbol is introduced since the nature of the string to be reduplicated is unknown. An alternative to reduplication is the following. We allow to concatenate two strings  $\vec{x}$  and  $\vec{y}$  *on condition that they are identical*. Thus, the formation of the plural in Malay can be expressed in two ways: by a reduplication rule, using a function

$$(1.35) \quad r(x) = x \hat{ } x$$

or by partial concatenation, using the function

$$(1.36) \quad c(x, y) = \begin{cases} x \hat{ } y & \text{if } x = y \\ \text{undefined} & \text{else} \end{cases}$$

The advantage of the latter is that every occurrence of a letter can be uniquely traced back to a leaf. The disadvantage is that it creates too many substitution classes.<sup>1</sup> Apart from that it is hard to distinguish this approach from the one based on duplication, the more so since the rule is completely general, and the categories will anyway turn out to be eliminable from the formulation of a grammar.

What the principle does *not* allow is the addition of any concretely specified symbol. For example, it may not say: “add an s at the end”. This must be represented alternatively as a binary rule concatenating the string with  $\vec{x}$ . Again,

---

<sup>1</sup>If we look at this rule in combination with semantics (anticipating the next chapter) we find that the reduplication approach will form the plural in the semantics by performing the step from property of individuals to properties of sets of individuals. The partial concatenation approach however makes the plurals appear more like *dvandva*-compounds. The idea is that in the Malay plural noun anak-anak ‘children’, we get the plural meaning from extrapolating a *dvandva* from ‘child’ and ‘child’ rather than (the more natural) *dvandva* formed from different parts.

requiring this we do not so much restrict *what* can be done but rather *how* it can be done.

Now we turn to the question of the alphabet and the nature of the underlying strings. Here, I admit, no unique and satisfying answer can be given. Two extremes exist: on the one hand we have alphabetic systems which are more or less sound based (with complications of their own kind). On the other we have ideographic systems like Chinese, which make a single letter correspond (again more or less) with a morpheme. Chinese presents a good example of the predicament we are facing: if we base our analysis on the *sound* form then there are about 100 letters (vowels in four tones plus consonants), or maybe somewhat more, given that pauses and intonation contours must be taken on board as well. If, however, we base our analysis on the alphabet of *characters* then we have an alphabet of up to 60000 ‘letters’. The question that naturally arises is this: which of the two should we choose? In principle, it seems, we should be able to do both, but writing systems can be so artificial that it seems we ought to exclude some of the writing systems.<sup>2</sup> But even if we do, the sound based approach presents difficulties of its own. One is that the notion of part is somewhat obscure. For example, we say that a string  $\vec{x}$  is *part* of a string  $\vec{y}$  if it is a subword. Thus, we may for example say that *eel* is part of *reel*, or *ice* is part of *rice*. If we apply our substitution tests, however, we get quite a bizarre picture of the language. Thus, we would like to apply substitution only to constituents, or, as we have said above, study those strings (or sequences) that can be substituted for a single letter. If *letter* can be equated with *morpheme*, we would get a far more interesting grammar from our substitution tests than if we insisted on sounds (or alphabetic characters). The disadvantage of the method is that it presupposes what it ought to reveal: the primitive parts. However, as we shall see in the next chapter, the notion of a morpheme makes perfect sense, because the alphabetic characters are in fact *not* the most elements, but the morphemes.

In stratificational based linguistics we actually pursue *both* analyses at once. There are various strata at which we have structure. Such frameworks have been pursued among other by [Lamb, 1966] and [Mel’čuk, 2000]. In our view the various levels are mostly epiphenomenal, and can be reconstructed on the basis of

---

<sup>2</sup>There was a way to write in Japan that used only Chinese characters and even Chinese word order. The characters were augmented with numbers so that one knew in which way to read the characters. Now, not only do the characters come out differently (the character for mountain is read *yama* in Japanese), but they are also arranged according to Chinese syntax.

the language itself.

Even if all this granted, we still face a number of problems. Suppose, for example, that our language is based on morphemes, which are the letters of our alphabet. Then, by our principles above, these letters must surface in our strings (or sequences of sounds). It follows that morphemes are sequences of characters of the alphabet. If so, we must address exceptions to strict concatenation. I mention here as representatives: final devoicing (as found in Russian and German, for example), vowel harmony (as found, say, in Finnish, Hungarian and Turkish), consonant lenition in Welsh, consonant gradation in Sami. Let us discuss the first case. Final devoicing is a process that turns any consonant in the coda of a syllable into a voiceless consonant. For example, there are two nouns in German, *Rad* [ʁa:t] ‘bicycle’, and *Rat* [ʁa:t] ‘council’. They sound exactly the same. On the other hand, their genitive, *Rades* [ʁa:dəs] and *Rates* [ʁa:təs], do not. The reason is that the rules of segmentation put the stop into the onset of the next syllable, where it does not undergo devoicing. If we base ourselves on the written forms, no problem. The sounds however do pose a problem. What can be the solution?

One solution ultimately rests on the distinction between complete and incomplete forms. Suppose that the base form comes without word end markers. So they would be [ʁa:d] and [ʁa:t], respectively. Now, when we attempt to pronounce such a word, we must speak it isolation, so we add a word boundary marker to its left and right: [#ʁa:d#] and [#ʁa:t#]. After that, there is a process that will produce the required form. This solution does explain the different outcomes, but it falls short of complying with the Principle of Preservation. This applies to all other phenomena listed above, which is why we have mentioned them. We shall therefore relax this principle a little bit. We shall assume that it is not the actual surface forms that must be preserved but a more abstract form.

If we left matters at that we would basically remove all restrictions. We need to restrict the abstraction. This is done as follows. We operate now with two levels: SP (the surface phonological level) and DP (the deep phonological level). Each of the levels uses the same alphabet (tentatively). The principles apply only to DP. The actual strings of SP are obtained by applying a finite state transducer. In other terms, the relation between DP and SP is *regular* (see [Kracht, 2003] for definitions and discussion). To account for German devoicing, we assume that in DP no devoicing applies. The relation to SP, however, is such that whatever consonant happens to be syllable final, is devoiced. This can be achieved using a finite state transducer.



## Chapter 2

# Compositionality and Leibniz' Principle

Two principles will be introduced in this chapter: both concern the relationship of strings with their meanings. To be able to formulate them properly, we shall have to introduce interpreted languages and grammars.

### 2.1 Interpreted Languages and Grammars

We assume the setup of the previous section. As we have said, objects of a language are sequences of strings over some alphabet (modulo a regular transduction). To avoid having to talk about the exact nature of syntactic objects, we assume that they come from a set  $E$ . This can be, for example,  $A^*$ , but different choices are possible (and often necessary).

To differentiate the previous type of language from the interpreted languages defined below we shall call languages as defined in Chapter 1 **string languages** (though in fact we have said that the exponents can be sequences of strings).

**Definition 2.1** *Let  $E$  and  $M$  be sets (of exponents and meanings, respectively). A*

set  $L \subseteq E \times M$  is called an *interpreted language over E*. The set

$$(2.1) \quad s(L) := \{\vec{x} : \text{there is } m \in M : \langle \vec{x}, m \rangle \in L\}$$

the *string language of L*. The set

$$(2.2) \quad m(L) := \{m : \text{there is } e \in E : \langle e, m \rangle \in L\}$$

is the *expressive power of M*.

In this connection we speak of the pair  $\sigma = \langle e, m \rangle$  as a **sign**.  $e$  is the **exponent** of  $\sigma$  and  $m$  its **denotatum**. We define the functions  $\varepsilon$  and  $\mu$  by

$$(2.3) \quad \varepsilon(\langle e, m \rangle) := e, \quad \mu(\langle e, m \rangle) := m$$

Thus a language is generally defined to be a set of signs; that a sign is seen here just as a binary relation and not ternary one (see Section 2.2) is mainly due to the fact that form and meaning are the most obvious components of it. The exponent can be seen, heard or touched (think of Braille letters), and the meaning—although somewhat hard to establish in exact detail—is what makes language a symbolic system. With this definition we also return to the roots. The definition of a sign pairing form and meaning is due to [Saussure, 1965]. De Saussure speaks of **signifiant** and **signifié**, rather than of *exponent* and *meaning*. The straightforward generalisation of the definition of grammar would be the following.

**Definition 2.2** Let  $A$  be an alphabet and  $M$  a set. An *interpreted grammar* is a pair  $G = \langle \Omega, \mathcal{J} \rangle$  where  $\Omega$  is a finite signature and  $\mathcal{J}$  a function that assigns to a symbol  $f \in F$  a partial function  $(E \times M)^{\Omega(f)} \hookrightarrow (E \times M)$ .

To put it somewhat more simply, given  $E$  and  $M$ , the set  $\Sigma := E \times M$  is the **space of signs**. If  $f$  is a function symbol,  $\mathcal{J}(f)$  is a partial  $n$ -ary function on  $\Sigma$ . However, for some theoretical reasons I would like to propose a somewhat more general definition. Notice namely that the output of  $\mathcal{J}(f)$  applied to some signs is a pair. So we can represent the function as a *pair* of functions  $f^\varepsilon$  and  $f^\mu$ :

$$(2.4) \quad \mathcal{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1}) = \langle f^\varepsilon(\sigma_0, \dots, \sigma_{\Omega(f)-1}), f^\mu(\sigma_0, \dots, \sigma_{\Omega(f)-1}) \rangle$$

We may revise the previous definition to the following.

**Definition 2.3** Let  $A$  be an alphabet and  $M$  a set. An **interpreted grammar**<sup>×</sup> is a triple  $G = \langle \Omega, \mathcal{J}_\varepsilon, \mathcal{J}_\mu \rangle$  where  $\Omega$  is a finite signature and  $\mathcal{J}_\varepsilon$  and  $\mathcal{J}_\mu$  functions that assign to a mode  $f$  a partial function  $\mathcal{J}_\varepsilon(f) : (E \times M)^{\Omega(f)} \hookrightarrow E$  and  $\mathcal{J}_\mu(f) : (E \times M)^{\Omega(f)} \hookrightarrow M$ .

The added <sup>×</sup> reminds us that this really is a different concept. If  $G = \langle \Omega, \mathcal{J}_\varepsilon, \mathcal{J}_\mu \rangle$  is an interpreted grammar<sup>×</sup> then put

$$(2.5) \quad \mathcal{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1}) = \langle \mathcal{J}_\varepsilon(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1}), \mathcal{J}_\mu(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1}) \rangle$$

Then  $G^\times := \langle \Omega, \mathcal{J} \rangle$  is an interpreted grammar. Conversely, an interpreted grammar  $G = \langle \Omega, \mathcal{J} \rangle$  can be turned into an interpreted grammar<sup>×</sup>. Simply define the interpretations in the following way:

$$(2.6) \quad \begin{aligned} \mathfrak{J}_\varepsilon(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1}) &:= \varepsilon(\mathfrak{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1})) \\ \mathfrak{J}_\mu(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1}) &:= \mu(\mathfrak{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1})) \end{aligned}$$

Then  $G^\ddagger := \langle \Omega, \mathcal{J}_\varepsilon, \mathcal{J}_\mu \rangle$  is an interpreted grammar. However, there are typically several grammars<sup>×</sup> that reduce to the same grammar. We call such a grammar<sup>×</sup> **balanced**. Notice that if the functions satisfy (2.6) then the domain of  $\mathfrak{J}_\mu$  and  $\mathfrak{J}_\varepsilon$  is the same: they are defined on exactly the same sequences of signs. It is however possible in an interpreted grammar<sup>×</sup> that  $f^\varepsilon$  and  $f^\mu$  have different domains. In that case the domain of  $f^\varepsilon \times f^\mu$  is the intersection of the two domains:

$$(2.7) \quad \text{dom}(f^\varepsilon \times f^\mu) = \text{dom}(f^\varepsilon) \cap \text{dom}(f^\mu)$$

An example will be given below.

The terminology of Section 1.1 for grammars is taken over unchanged. For example, the definition of analysis term is the same (it involves only the underlying signature) and the interpretation is defined inductively in the same manner. The reason is that the same signature can be applied to generate string languages, and to generate interpreted string languages (and even more complex languages, which we shall consider below in Section 2.2). It just depends on the function  $\mathcal{J}$  what types of objects are generated. For example, given an interpreted grammar  $G = \langle \Omega, \mathcal{J} \rangle$ , we define the interpretation of a constant term  $t$  by induction as follows:

$$(2.8) \quad \iota_G(f s_0 s_1 \dots s_{\Omega(f)-1}) := \mathcal{J}(f)(\iota_G(s_0), \iota_G(s_1), \dots, \iota_G(s_{\Omega(f)-1}))$$

We use also the following notation. For terms  $t$  we let  $t^e$  be the exponent of  $\iota(t)$  and  $t^m$  the meaning. A term is **semantically definite** if  $t^m$  exists; and **orthographically definite** if  $t^e$  exists. Terms that contain variables are interpreted as partial functions from  $S^{\mathbb{N}} \hookrightarrow S$ , where  $S$  is the space of signs, here  $E \times M$ . Given a sequence  $\langle \sigma_0, \sigma_1, \dots \rangle$  of signs  $\iota(t)$  computes the value of  $t$  where for every  $i \in \mathbb{N}$ ,  $x_i$  is interpreted as  $\sigma_i$ .

**Example 8.**  $A := \{\mathbf{0}, 1, +, -, (, ), =\}$ .  $F := \{f_0, f_1, f_2, f_3, f_4, f_5, f_6\}$ .  $\Omega(f_0) := \Omega(f_1) := 0$ ,  $\Omega(f_2) := \Omega(f_3) := 2$ ,  $\Omega(f_4) := \Omega(f_5) := \Omega(f_6) := 2$ .  $\vec{x}$  is **binary** if it only contains  $\mathbf{0}$  and  $1$ ;  $\vec{x}$  is a **term** if it does not contain  $=$ .

$$\begin{aligned}
 \mathcal{J}(f_0)(\mathbf{0}) &:= \langle \mathbf{0}, 0 \rangle \\
 \mathcal{J}(f_1)(\mathbf{1}) &:= \langle \mathbf{1}, 1 \rangle \\
 \mathcal{J}(f_2)(\langle \vec{x}, n \rangle) &:= \begin{cases} \langle \vec{x}\mathbf{0}, 2n \rangle & \text{if } \vec{x} \text{ is binary} \\ \text{undefined} & \text{else} \end{cases} \\
 \mathcal{J}(f_3)(\langle \vec{x}, n \rangle) &:= \begin{cases} \langle \vec{x}\mathbf{1}, 2n + 1 \rangle & \text{if } \vec{x} \text{ is binary} \\ \text{undefined} & \text{else} \end{cases} \\
 (2.9) \quad \mathcal{J}(f_4)(\langle \vec{x}, n \rangle, \langle \vec{y}, m \rangle) &:= \begin{cases} \langle (\vec{x}+\vec{y}), n + m \rangle & \text{if } \vec{x}, \vec{y} \text{ are terms} \\ \text{undefined} & \text{else} \end{cases} \\
 \mathcal{J}(f_5)(\langle \vec{x}, n \rangle, \langle \vec{y}, m \rangle) &:= \begin{cases} \langle (\vec{x}-\vec{y}), n - m \rangle & \text{if } \vec{x}, \vec{y} \text{ are terms} \\ \text{undefined} & \text{else} \end{cases} \\
 \mathcal{J}(f_6)(\langle \vec{x}, n \rangle, \langle \vec{y}, m \rangle) &:= \begin{cases} \langle \vec{x}=\vec{y}, \top \rangle & \text{if } \vec{x}, \vec{y} \text{ are terms and } m = n \\ \langle \vec{x}=\vec{y}, \perp \rangle & \text{if } \vec{x}, \vec{y} \text{ are terms and } m \neq n \\ \text{undefined} & \text{else} \end{cases}
 \end{aligned}$$

The signs that this grammar generates are of the following form. They are either strings of 0s and 1s, paired with the number that they represent as binary numbers. Or they are terms, interpreted in the usual way; or they are equations between two such terms. A single numeral expression is also a term. An equation is either true (in which case it is interpreted by  $\top$ ) or false (in which case it is interpreted by  $\perp$ ).

♣

**Example 9.** We shall now define some unbalanced interpreted grammar<sup>×</sup> that defines the same interpreted grammar as the previous example.

$$\begin{aligned}
& \mathcal{K}_\varepsilon(f_0)() := \mathbf{0} \\
& \mathcal{K}_\varepsilon(f_1)() := \langle 1, 1 \rangle \\
& \mathcal{K}_\varepsilon(f_2)(\langle \vec{x}, n \rangle) := \begin{cases} \vec{x}\mathbf{0} & \text{if } \vec{x} \text{ is binary} \\ \text{undefined} & \text{else} \end{cases} \\
& \mathcal{K}_\varepsilon(f_3)(\langle \vec{x}, n \rangle) := \begin{cases} \vec{x}\mathbf{1} & \text{if } \vec{x} \text{ is binary} \\ \text{undefined} & \text{else} \end{cases} \\
(2.10) \quad & \mathcal{K}_\varepsilon(f_4)(\langle \vec{x}, n \rangle, \langle \vec{y}, m \rangle) := \begin{cases} (\vec{x}+\vec{y}) & \text{if } \vec{x}, \vec{y} \text{ are terms} \\ \text{undefined} & \text{else} \end{cases} \\
& \mathcal{K}_\varepsilon(f_5)(\langle \vec{x}, n \rangle, \langle \vec{y}, m \rangle) := \begin{cases} (\vec{x}-\vec{y}) & \text{if } \vec{x}, \vec{y} \text{ are terms} \\ \text{undefined} & \text{else} \end{cases} \\
& \mathcal{K}_\varepsilon(f_6)(\langle \vec{x}, n \rangle, \langle \vec{y}, m \rangle) := \begin{cases} \vec{x}=\vec{y} & \text{if } \vec{x}, \vec{y} \text{ are terms and } m = n \\ \langle \vec{x}=\vec{y}, \perp \rangle & \text{if } \vec{x}, \vec{y} \text{ are terms and } m \neq n \\ \text{undefined} & \text{else} \end{cases}
\end{aligned}$$

For the semantic functions we choose

$$\begin{aligned}
& \mathcal{K}_\mu(f_0)() := 0 \\
& \mathcal{K}_\mu(f_1)() := 1 \\
& \mathcal{K}_\mu(f_2)(\langle \vec{x}, n \rangle) := 2n \\
& \mathcal{K}_\mu(f_3)(\langle \vec{x}, n \rangle) := 2n + 1 \\
(2.11) \quad & \mathcal{K}_\mu(f_4)(\langle \vec{x}, n \rangle, \langle \vec{y}, m \rangle) := n + m \\
& \mathcal{K}_\mu(f_5)(\langle \vec{x}, n \rangle, \langle \vec{y}, m \rangle) := n - m \\
& \mathcal{K}_\mu(f_6)(\langle \vec{x}, n \rangle, \langle \vec{y}, m \rangle) := \begin{cases} \top & \text{if } m = n \\ \perp & \text{if } m \neq n \end{cases}
\end{aligned}$$

For the interpreted grammar  $G = \langle \Omega, \mathcal{K}_\varepsilon, \mathcal{K}_\mu \rangle$  we find that  $G^\times = \langle \Omega, \mathfrak{F} \rangle$ . However, it does not satisfy the equations (2.6). For if it did, the semantic function would have to be partial. For example,  $\mu(\beta(f_2)(\langle (1+1), 2 \rangle))$  is undefined. On the other hand,  $\mathfrak{R}_\mu(\langle (1+1), 2 \rangle) = 4$ , since  $\mathcal{K}_\mu$  does not look at the exponent. Notice that the semantic functions are total. Notice also that they do not depend on the exponent, so they can be further simplified. This will be discussed in detail in Section 2.3. ♣

**Exercise 3.** Previously, we interpreted the modes  $f_2$  and  $f_3$  by the string functions  $\vec{x} \mapsto 0 \hat{\ } \vec{x}$  and  $\vec{x} \mapsto 1 \hat{\ } \vec{x}$ . Show that it is impossible to use the meaning functions as above with these string functions.

**Exercise 4.** (Continuing the previous exercise.) Give a grammar that generates the language of equations using the string functions above. (Evidently, the functions on meanings must be quite different.)

**Exercise 5.** Say that  $L$  is **unambiguous** if for every  $\langle e, m \rangle, \langle e, m' \rangle \in L$  we have  $m = m'$ . Suppose that there exists an interpreted grammar<sup>x</sup>  $\langle \Omega, \mathcal{J}_e, \mathcal{J}_\mu \rangle$  for  $L$ . Show that there exists an interpreted grammar<sup>x</sup> such that  $\mathcal{J}_\mu(f)$  is total for every mode  $f$ .

**Exercise 6.** Give an example of a language that is unambiguous and has an interpreted grammar<sup>x</sup> but none where  $\mathcal{J}_e$  is total for every mode  $f$ .

## 2.2 Do We Need Categories?

Following the tradition in linguistics, I have assumed in [Kracht, 2003] that signs are triples  $\sigma = \langle e, c, m \rangle$ , with  $e$  the exponent,  $m$  the meaning, and  $c$  the **category** of  $\sigma$ . This is in line with [Pollard and Sag, 1994], [Mel'čuk, 2000], not to mention Categorical Grammar, for which categories are essential, and even recent LFG, which assumes a level of m-structures in addition to c-structure (syntax) and f-structure (semantics) and even a-structure (to deal with argument handling), see [Falk, 2001]. However, from an abstract viewpoint we must ask if categories are really necessary. After all, each level that is added introduces new degrees of freedom and new ways to outplay restrictions in other levels. And, to add to that, the categories are actually not directly observable. [Chomsky, 1993] assumes that language pairs form with meaning. Whatever this says in the practice of generative grammar (and in practice the syntactic categories reappear in the form part), the initial hypothesis is the same: start with a set of signs that contain only form and meaning. This section will prove that in case the set of signs contains only finitely many categories their addition is insignificant.

The formal details are as follows. Say a **c-sign** is a triple  $\gamma = \langle e, c, m \rangle$ . The space of c-signs is given as a product  $E \times C \times M$ . A **c-language** is a set of c-signs. Put

$$(2.12) \quad H(\gamma) := \langle e, m \rangle$$

A **c-grammar** consists in a signature of modes  $\langle F, \Omega \rangle$  plus an interpretation function  $\mathcal{C}$ , which for given  $f$  returns a partial function  $(E \times C \times M)^{\Omega(f)} \hookrightarrow (E \times C \times M)$ . The **c-language** of  $G$ ,  $L(G)$ , is the set of c-signs generated by this grammar. This is defined inductively in the usual way. Now, given  $L = L(G)$ , the  $H$ -image is

$$(2.13) \quad H[L] := \{\langle e, m \rangle : \text{there is } c \in C : \langle e, c, m \rangle \in L\}$$

**Theorem 2.4** *Let  $G = \langle \mathcal{C}, \Omega \rangle$  be a c-grammar such that  $L = L(G) \subseteq E \times C \times M$  for some finite  $C$ . Then there exists an interpreted grammar  $K$  such that  $L(K) = H[L]$ .*

**Proof.** Let  $\langle F, \Omega \rangle$  be the signature of  $G$ . For a natural number  $i$  let  $F_i$  be the set of  $f$  such that  $\Omega(f) = i$ . Define

$$(2.14) \quad \begin{aligned} F_0^+ &:= F_0 \\ F_1^+ &:= \{f_c : f \in F_1, c \in C\} \\ F_2^+ &:= \{f_{c,c'} : f \in F_2, c, c' \in C\} \\ &\dots \end{aligned}$$

As for the signature, we put

$$(2.15) \quad \Omega^+(f_c) := \Omega(f)$$

We define the actions of the functions over this signature.

$$(2.16) \quad \begin{aligned} \mathcal{J}(f_{c_0, c_1, \dots, c_{n-1}})(\langle e_0, m_0 \rangle, \langle e_1, m_1 \rangle, \dots, \langle e_{n-1}, m_{n-1} \rangle) \\ := H(\mathcal{C}(f)(\langle e_0, c_0, m_0 \rangle, \langle e_1, c_1, m_1 \rangle, \dots, \langle e_{n-1}, c_{n-1}, m_{n-1} \rangle)) \end{aligned}$$

This can also be written as follows. Given that  $\sigma_i = \langle e_i, c_i, m_i \rangle$ :

$$(2.17) \quad \begin{aligned} \mathcal{J}(f_{c_0, c_1, \dots, c_{n-1}})(H(\sigma_0), H(\sigma_1), \dots, H(\sigma_{n-1})) \\ := H(\mathcal{C}(f)(\sigma_0, \sigma_1, \dots, \sigma_{n-1})) \end{aligned}$$

Here the left hand side is defined iff the right hand side is; and in that case the left hand side is defined to be whatever the right hand side is. This defines the grammar  $K := \langle \Omega, \mathcal{J} \rangle$ .

We shall show that  $L(K) = H[L]$ . First:  $L(K) \supseteq H[L(G)]$ . To this effect, let  $\sigma \in L(G)$ . We show that  $H(\sigma) \in L(K)$ . By assumption, there is a term  $t$  in

the signature  $\Omega$  such that  $\iota_G(t) = \sigma$ . We shall construct a term  $t^+$  by induction on  $t$  and show that  $\iota_K(t^+) = H(\iota_G(t)) = H(\sigma)$ . Base case.  $t = f$ , where  $f$  is constant. Then  $f^+ := f$ . Now,  $\iota_K(f^+) = H(\iota_G(f))$ , by construction. Inductive case.  $t = f s_0 s_1 \cdots s_{n-1}$ .  $\Omega(f)n > 0$ . Let  $\iota_G(s_i) = \langle e_i, c_i, m_i \rangle$ . By induction hypothesis, for every  $i < n$  there is a term  $s_i^+$  such that  $\iota_K(s_i^+) = H(\iota_G(s_i))$ . Then  $\mathcal{C}(f)$  is defined on the  $\iota_G(s_i)$ , and therefore  $\mathcal{J}(f_{c_0, c_1, \dots, c_{n-1}})$  is defined on  $\langle e_i, m_i \rangle = \iota_K(s_i^+)$  and yields the value

$$\begin{aligned}
 \iota_K(t^+) &= \mathcal{J}(f_{\vec{c}})(\iota_K(s_0^+), \iota_K(s_1^+), \dots, \iota_K(s_{n-1}^+)) \\
 &= H(\mathcal{C}(f)(\iota_G(s_0), \iota_G(s_1), \dots, \iota_G(s_{n-1}))) \\
 (2.18) \quad &= H(\iota_G(t)) \\
 &= H(\sigma)
 \end{aligned}$$

Second:  $L(K) \subseteq H[L]$ . Let  $\sigma \in L(K)$ . Then there is a term  $t$  such that  $\iota_K(t) = \sigma$ . Put  $t^-$  as follows:

$$(2.19) \quad (f_{\vec{c}} s_0 \cdots s_{\Omega(f)-1})^- := f s_0^- s_1^- \cdots s_{\Omega(f)-1}^-$$

We shall show that  $H(\iota_G(t^-)) = \iota_K(t)$ ; for then put  $\gamma := \iota_G(t^-)$ . It follows that  $H(\gamma) = \sigma$ . The remaining proof is by induction on  $t$ . Base case.  $\Omega(f_{\vec{c}}) = 0$ . In this case  $H(\iota_G(t^-)) = \iota_K(t)$ , by assumption. Inductive case.  $n := \Omega(f) > 0$ . Let  $\iota_G(s_i^-) = c_i$  and  $\vec{c} = \langle c_0, c_1, \dots, c_{n-1} \rangle$ . Then, using (2.17):

$$\begin{aligned}
 H(\iota_G(t^-)) &= H(\iota_G(f s_0^- \cdots s_{n-1}^-)) \\
 &= H(\mathcal{C}(f)(\iota_G(s_0^-), \dots, \iota_G(s_{n-1}^-))) \\
 (2.20) \quad &= \mathcal{J}(f_{\vec{c}})(H(\iota_G(s_0^-)), H(\iota_G(s_1^-)), \dots, H(\iota_G(s_{n-1}^-))) \\
 &= \mathcal{J}(f_{\vec{c}})(\iota_K(s_0), \iota_K(s_1), \dots, \iota_K(s_{n-1})) \\
 &= \iota_K(t)
 \end{aligned}$$

This had to be shown. □

We shall write  $H(G)$  for the grammar  $K$ , for future reference. Notice that the base cases are actually redundant in both parts; they are covered by the induction step!

This result is of some significance. It says that the categories are redundant. More precisely, they can be removed from the signs at the cost of introducing more modes of composition. The proof is completely general; it uses no assumptions on the grammar. This applies to CFGs, but there are other cases too. Categorical

grammars in principle use an infinite number of categories. However, the categories are not always needed. It may well be that the lexicon allows to produce only finitely many categories in any case. Such is the case in the Ajdukiewicz-Bar Hillel Calculus. The Lambek-Calculus is different in that we can create and use infinitely many categories (for example, if we have the product then we can form arbitrarily long categories). However, given that the Lambek-Calculus yields a context free language (see [Pentus, 1997]) it therefore enjoys a formulation using no categories whatsoever, by the above theorem.

It is worth pointing out why this theorem is actually not trivial. Suppose that a language has nouns and verbs, and that these word classes are morphologically distinct. Suppose further that there are roots that can be used as nouns and verbs. English is such a language. Here are examples: *dust*, *walk*, *leak*, and so on, are examples of words that can be either nouns or verbs. Dictionaries see the matter as follows: the word *leak* can be both a noun and a verb; if it is a noun it means something, say  $X$ , if it is a verb it means something else, say  $Y$ . Thus, dictionaries use categories; they say that the language contains two signs:  $\langle \text{leak}, n, X \rangle$  and  $\langle \text{leak}, v, Y \rangle$ . If we eliminate the categories, we are left with the signs  $\langle \text{leak}, X \rangle$  and  $\langle \text{leak}, Y \rangle$ . It seems that vital information is lost, namely that *leak* means  $X$  *only if it is a noun*, and likewise that it means  $Y$  *only if it is a verb*. On the other hand, we still know that *leak* means  $X$  and  $Y$ . If we perform the construction above, the following will happen. The function that forms the past tense applies to the sign  $\langle \text{leak}, v, Y \rangle$  but not to the sign  $\langle \text{leak}, n, X \rangle$ . It is the interpretation of some mode  $f$ . This mode is now replaced by a mode  $f_v$ , which takes as input only the sign  $\langle \text{leak}, Y \rangle$  and forms the sign  $\langle \text{leaked}, \text{past}'(Y) \rangle$ . It is not defined on  $\langle \text{leak}, X \rangle$ . Similarly the other functions are described.

Notice that the elimination of categories results in a redistribution of grammatical knowledge. The morphological (or syntactic) information is placed elsewhere. It used to be encoded in the categories of the signs. Now it is encoded in the domain of the newly introduced functions. For example, the domain of the function  $f_v$  forming the past tense of verbs is the set of pairs  $\langle \vec{x}, X \rangle$  where  $\vec{x}$  is a root and  $X$  the verbal meaning of that root. It is undefined on  $\langle \vec{y}, X \rangle$  if  $\vec{y}$  cannot be a verbal root or otherwise does not have the meaning  $X$ ; it is not defined on  $\langle \vec{x}, Y \rangle$  if  $Y$  is not a meaning of the verbal root  $\vec{x}$ .

Although categories *can* be eliminated, this does not mean that they *should* be eliminated. One reason is purely practical: in evaluating a term, the computation may be much easier if we carried along category information, since the categories

can be made to fit the partial nature of the functions. This is quite clear in Categorical Grammar, for example. To see whether a mode applies to certain signs it is enough to check the categories. If we used the above definition, we would have to recompute the category of the signs over and over. Additionally, we shall show below that the elimination of categories can have the effect of removing desirable properties from the grammar. Hence it may be desirable to keep the format in the usual way; it is however essential to know that categories are theoretically redundant.

### 2.3 Compositionality and Independence

In this section we shall look at the independence of the components of a sign of each other. We have so far assumed that the modes are interpreted as functions on signs. As such they have the form

$$(2.21) \quad \mathcal{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1}) = \langle f^\varepsilon(\sigma_0, \dots, \sigma_{\Omega(f)-1}), f^\mu(\sigma_0, \dots, \sigma_{\Omega(f)-1}) \rangle$$

where  $f^\varepsilon$  and  $f^\mu$  are partial functions from signs to exponents and meanings, respectively. Notice the slight notational inconsistency:  $f^\varepsilon$  is a function from signs to exponents, while  $f$  is just a mode, whose interpretation in  $G$  we denote by  $\mathcal{J}(f)$ . The above equation is a partial equation; that means, unless stated otherwise, that one side is defined iff the other is, and that they are equal if defined. A more concise formulation of the above equation is the following functional equation:

$$(2.22) \quad \mathcal{J}(f) = f^\varepsilon \times f^\mu$$

Then for the functions on the right we have

$$(2.23) \quad f^\varepsilon = \varepsilon \circ \mathcal{J}(f), \quad f^\mu = \mu \circ \mathcal{J}(f).$$

On the other hand, we could opt for the broader concept of an interpreted grammar<sup>x</sup>, and put

$$(2.24) \quad f^\varepsilon := \mathcal{J}^\varepsilon(f), \quad f^\mu := \mathcal{J}^\mu(f)$$

in which case (2.23) no longer holds. Although we do not always mention this fact, the reader is asked to be aware of the possibility of using a grammar<sup>x</sup> in place of a grammar, which may open more possibilities to define grammars.

Before we enter into technicalities I should point out that there are two senses in which these equations can be required to hold. I call the first the **intensional sense**. The equations are valid as stated above. That means that the functions specified are valid even if the relevant functions are defined on signs not in the language. The **extensional sense** requires that the equations only hold for the language of the grammar. Mathematically, this would be expressed as

$$(2.25) \quad \mathcal{J}(f) \cap L(G) = (f^\varepsilon \times f^\mu) \cap L(G)$$

These two viewpoints really are different. It is assumed that the grammatical formation rules are more general; they may be applied to words (and meanings) that do not exist. For example, we may introduce into any language new words. What we find is that more often than not the morphological rules know how to deal with them. If the rules were just defined on the language as it is, we would have to artificially extend the interpretation of the modes as soon as new entries get introduced into the lexicon. Thus, we assume with some justification that the functions above are possibly defined on signs outside of the language generated by the grammar. Nevertheless we shall study the behaviour of the functions in the intensional sense. This is because it is easy to return to the extensional sense by restricting the original functions to  $L(G)$ . Formally, this may be expressed as follows. We say that  $G' = \langle \Omega, \mathcal{J}' \rangle$  is an **extensional variant** of  $G = \langle \Omega, \mathcal{J} \rangle$  if  $L(G') = L(G)$  and for every mode  $f$ ,  $\mathcal{J}'(f) \cap L(G) = \mathcal{J}(f) \cap L(G)$ . Extensional variants cannot be distinguished from each other by looking at the language they generate; but they might be distinguishable by introducing ‘nonce signs’.

Let’s return to the equation (2.21) above. I shall rewrite it as follows:

$$(2.26) \quad \begin{aligned} & \mathcal{J}(f)(\langle e_0, m_0 \rangle, \dots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1} \rangle) \\ &= \langle f^\varepsilon(\langle e_0, m_0 \rangle, \dots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1} \rangle), \\ & \quad f^\mu(\langle e_0, m_0 \rangle, \dots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1} \rangle) \rangle \end{aligned}$$

We say that a grammar is compositional if  $f^\mu$  does not depend on the  $e_i$ . This can be restated as follows.

**Definition 2.5** *An interpreted grammar  $G = \langle \Omega, \mathcal{J} \rangle$  is **compositional** if for every mode  $f$ ,  $f^\mu$  does not depend on the exponents of the signs.  $G$  is **extensionally compositional** if it has an extensional variant that is compositional. An interpreted language  $L$  is **compositional** if there is a compositional interpreted grammar  $G$  such that  $L = L(G)$ .*

If  $G$  is only extensionally compositional then for every mode  $f$  there exists a partial function  $\hat{f}^\mu : M^{\Omega(f)} \hookrightarrow M$  such that

$$(2.27) \quad \mu(\mathcal{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1})) = \hat{f}^\mu(\mu(\sigma_0), \dots, \mu(\sigma_{\Omega(f)-1}))$$

with the right side defined if the left hand side is (but not conversely). If  $G$  is compositional then (2.26) becomes

$$(2.28) \quad \mu(\mathcal{J}(f)(\langle e_0, m_0 \rangle, \dots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1} \rangle)) = f_*^\mu(m_0, \dots, m_{\Omega(f)-1})$$

where  $f_*^\mu$  is defined by

$$(2.29) \quad f_*^\mu(m_0, \dots, m_{\Omega(f)-1}) := f^\mu(\langle e, m_0 \rangle, \langle e, m_1 \rangle, \dots, \langle e, m_{\Omega(f)-1} \rangle)$$

Here,  $e$  is chosen arbitrarily. Since by assumption  $f^\mu$  does not depend on the exponents, any choice will do. Another definition is to take the full image of the function  $f$  under projection. Recall that an  $n$ -ary function  $g$  on signs is a subset of  $(E \times M)^{n+1}$ . For any such function put

$$(2.30) \quad \mu[g] := \{\langle \mu(\sigma_0), \dots, \mu(\sigma_n) \rangle : \langle \sigma_0, \dots, \sigma_n \rangle \in g\}$$

Then we may alternatively define  $f_*^\mu$  by

$$(2.31) \quad f_*^\mu := \mu[\mathcal{J}(f)]$$

Independence from the exponents guarantees that this is a function. We see here more explicitly that  $f_*^\mu$  is a partial function only on meanings. Suppose now that  $L$  is compositional; this means that there is a compositional grammar  $G$  such that  $L = L(G)$ . This means in turn that for every  $\sigma \in L$  there is a term  $t$  such that  $\sigma = \iota_G(t)$ . If  $t = f s_0 \dots s_{\Omega(f)-1}$  then the meaning of  $\iota_G(t)$  equals  $f_*^\mu(\mu(\iota_G(s_0)), \dots, \mu(\iota_G(s_{\Omega(f)-1})))$ , which is to say that, given that the  $\sigma_i$  are the parts of  $\sigma$ , the meaning of  $\sigma$  is the result of applying the function  $f^\mu$  to the meaning of its parts. However, notice that we have two senses of compositionality, the simple (intensional) and the extensional. For a language to be compositional we may require the existence of either an extensionally compositional grammar, or of a compositional grammar. For if an extensionally compositional grammar exists, there is a compositional variant, which by definition generates the same language.

Notice a further consequence. If  $G$  is extensionally compositional then we can produce an extensional variant in the following way. Put

$$(2.32) \quad \hat{f}^\varepsilon := (\varepsilon \circ \mathcal{J}(f)) \cap L(G)$$

This function is defined exactly in the signs of  $L(G)$ . Now take as  $\hat{f}_*^\mu$  any function extending  $f_*^\mu$ . (Here is an example: suppose we defined a function ‘past’ that computes the past tense of nouns in addition to verbs. Then though this is improper we would not notice since the function ‘past tense formation’ on the exponents rejects the input anyway.) A particular choice that we may take is  $\mu[\mathcal{J}(f)]$ . This is sufficient. Notice however that this may still be a partial function. Any function extending it will also do, but nothing less.

In and of itself this seems to capture the definition of compositionality. However, it presupposes a notion of a part and mode of composition. There are two ways to understand ‘part’ and ‘mode of composition’. We may simply say that it is the grammar that defines what is part of what and what is a mode. Or we may say that the notion of part is not arbitrary. Not every grammar implements a correct notion of ‘part of’. Not every grammar therefore uses a good notion of ‘mode of composition’. In [Kracht, 2003] I have put the restrictions into the definition of compositionality. Here I shall keep them separate.

There is a corresponding notion of *autonomy* which runs as follows.

**Definition 2.6** *An interpreted grammar  $G = \langle \Omega, \mathcal{J} \rangle$  is **autonomous** if for every mode  $f$  the function  $f^\varepsilon$  does not depend on the  $m_i$ .  $G$  is **extensionally autonomous** if it has an extensional variant that is autonomous. An interpreted language  $L$  is **autonomous** if there is an autonomous interpreted grammar  $G$  such that  $L = L(G)$ .*

Autonomy says that the exponent of a complex sign is the result of applying a certain function to the exponent of its parts, and that function depends only on the leading symbol of the analysis term. One consequence is that for every mode  $f$  there exists a partial function  $f_*^\varepsilon : E^{\Omega(f)} \hookrightarrow E$  such that

$$(2.33) \quad \varepsilon(\mathcal{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1})) = f_*^\varepsilon(\varepsilon(\sigma_0), \dots, \varepsilon(\sigma_{\Omega(f)-1}))$$

with the left and side defined if the right hand side is (but not conversely).

Finally, we say our language is *independent* if both syntax and semantics can operate independently from each other.

**Definition 2.7** *An interpreted grammar is **independent** if it is both compositional and autonomous; it is **extensionally independent** if it is both extensionally com-*

positional and extensionally autonomous. A language is **independent** if it has an independent grammar.

Thus  $G$  is independent if for every  $f$  there and the functions  $f_*^\varepsilon$  and  $f_*^\mu$  we have that for all  $\sigma_i = \langle e_i, m_i \rangle, i < n$ :

$$(2.34) \quad \mathcal{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1}) = \langle f_*^\varepsilon(e_0, \dots, e_{\Omega(f)-1}), f_*^\mu(m_0, \dots, m_{\Omega(f)-1}) \rangle$$

with the left hand side defined iff the right hand side is. Another formulation is as follows:

$$(2.35) \quad \mathcal{J}(f) = (f_*^\varepsilon \circ \overbrace{\langle \varepsilon, \dots, \varepsilon \rangle}^{\Omega(f)}) \times (f_*^\mu \circ \overbrace{\langle \mu, \dots, \mu \rangle}^{\Omega(f)})$$

or

$$(2.36) \quad \mathcal{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1}) \\ = \langle f_*^\varepsilon(\varepsilon(\sigma_0), \dots, \varepsilon(\sigma_{\Omega(f)-1})), f_*^\mu(\mu(\sigma_0), \dots, \mu(\sigma_{\Omega(f)-1})) \rangle$$

It may be thought that extensional independence follows from extensional autonomy and extensional compositionality. However, this is not so.

**Example 10.** We construct four different grammars to show that autonomy and compositionality are independent notions. Let  $A = \{1\}$ ,  $E = A^*$ ;  $M = \mathbb{N}$ . The signature is  $\{f_0, f_1, f_2\}$ , with  $f_0$  nullary and  $f_1$  and  $f_2$  both unary. We have  $\mathcal{J}(f_0) = \langle \varepsilon, 0 \rangle$ ; and

$$(2.37) \quad \mathcal{J}(f_1)(\langle \vec{x}, n \rangle) := \langle \vec{x} \hat{=} 1, n + 1 \rangle \\ \mathcal{J}(f_2)(\langle \vec{x}, n \rangle) := \begin{cases} \langle \vec{x} \hat{=} 1, n \rangle & \text{if } |\vec{x}| \geq n \\ \langle \vec{x}, n + 1 \rangle & \text{else} \end{cases}$$

This grammar generates the language  $D = \{\langle \vec{x}, n \rangle : n \leq |\vec{x}|\}$ , as is easily verified. Notice that the second clause of the definition is never used inside  $D$ . This grammar is not autonomous:  $\mathcal{J}(f_2)(\langle 1, 3 \rangle) = \langle 1, 4 \rangle$ , but  $\mathcal{J}(f_2)(\langle 1, 1 \rangle) = \langle 11, 1 \rangle$ . So to compute the exponent we need to know the meaning. It is not compositional either. For we have  $\mathcal{J}(f_2)(\langle 111, 3 \rangle) = \langle 1111, 3 \rangle$ , so to compute the meaning we need to know the exponent.

Consider the following variants of  $\mathcal{J}$ , which agree on  $f_0$  and  $f_1$  with  $\mathcal{J}$ :

$$(2.38) \quad \begin{aligned} \mathcal{J}^a(f_2)(\langle \vec{x}, n \rangle) &:= \begin{cases} \langle \vec{x} \hat{\ } 1, n \rangle & \text{if } |\vec{x}| \geq n \\ \langle \vec{x} \hat{\ } 1, n+1 \rangle & \text{else} \end{cases} \\ \mathcal{J}^c(f_2)(\langle \vec{x}, n \rangle) &:= \begin{cases} \langle \vec{x} \hat{\ } 1, n \rangle & \text{if } |\vec{x}| \geq n \\ \langle \vec{x} \hat{\ } 1 \hat{\ } 1, n \rangle & \text{else} \end{cases} \\ \mathcal{J}^{ac}(f_2)(\langle \vec{x}, n \rangle) &:= \langle \vec{x} \hat{\ } 1, n \rangle \end{aligned}$$

All of them only generate the language  $D$ . The grammar  $G^{ac} := \langle \Omega, \mathcal{J}^{ac} \rangle$  is autonomous and compositional (even independent).  $G^c = \langle \Omega, \mathcal{J}^c \rangle$  is independent but not autonomous. For we have  $\mu(\mathcal{J}^c(f_2)(\langle e, m \rangle)) = m$ , which is independent of  $e$ ; but we have  $\varepsilon(\mathcal{J}^c(f_2)(\langle 11, 2 \rangle)) = 111 = 1111 = \varepsilon(\mathcal{J}^c(f_2)(\langle 11, 3 \rangle))$ . Similarly we find that  $\langle G^a := \langle \Omega, \mathcal{J}^a \rangle$  is autonomous but not compositional. ♣

Finally, let us return to interpreted grammars<sup>x</sup>. It is straightforward to reformulate the definitions above for the general setting. In fact, the only thing that changes is that instead of using the definition (2.23) to define  $f^\varepsilon$  and  $f^\mu$  we now use (2.24). Now, if an i-grammar<sup>x</sup> is autonomous then it is possible to define a variant of the form:  $\langle \Omega, \mathcal{J}_\varepsilon^*, \mathcal{J}_\mu^* \rangle$  where  $\mathcal{J}_\varepsilon^*(f)$  is total. Namely, observe that there is a function  $g$  on exponents such that

$$(2.39) \quad \mathcal{J}_\varepsilon(f)(\vec{\sigma}) = g(e_0, \dots, e_{\Omega(f)-1})$$

Choose a total extension  $g^* \supseteq g$ .

$$(2.40) \quad \begin{aligned} \mathcal{J}_\varepsilon^*(f)(\vec{\sigma}) &:= g^*(e_0, \dots, e_{\Omega(f)-1}) \\ \mathcal{J}_\mu^*(f) &:= \mathcal{J}_\mu(f) \cap \text{dom}(\mathcal{J}_\varepsilon(f)) \end{aligned}$$

Then  $\mathcal{J}^*(f)(\vec{\sigma})$  is undefined iff  $\vec{\sigma} \notin \text{dom}(\mathcal{J}_\mu^*(f)) = \text{dom}(\mathcal{J}_\varepsilon(f)) \cap \text{dom}(\mathcal{J}_\mu(f))$ ; and if it is defined, then

$$(2.41) \quad \begin{aligned} \langle \mathcal{J}_\varepsilon^*(f)(\vec{\sigma}), \mathcal{J}_\mu^*(f)(\vec{\sigma}) \rangle &= \langle g^*(\vec{\sigma}), \mathcal{J}_\mu(\vec{\sigma}) \rangle \\ &= \langle g(\vec{\sigma}), \mathcal{J}_\mu(\vec{\sigma}) \rangle \\ &= \langle \mathcal{J}_\varepsilon(\vec{\sigma}), \mathcal{J}_\mu(\vec{\sigma}) \rangle \end{aligned}$$

**Example 11.** From a grammar we can essentially make two gramars<sup>x</sup>: one where all the exponent functions are total, and another where the semantic functions are total. With a bit of luck the first grammar is autonomous and the second

compositional. Here is an example. Let  $A = \{1\}$ ,  $E = A^*$ ;  $M = \mathbb{N}$ . The signature is  $\{f_0, f_1, f_2\}$ , with  $f_0$  nullary and  $f_1$  and  $f_2$  both unary. We have  $\mathcal{J}(f_0) = \langle \varepsilon, 0 \rangle$ ; and

$$(2.42) \quad \begin{aligned} \mathcal{J}(f_1)(\langle \vec{x}, n \rangle) &:= \langle \vec{x}^\wedge 1, n + 1 \rangle \\ \mathcal{J}(f_2)(\langle \vec{x}, n \rangle) &:= \begin{cases} \langle \vec{x}^\wedge 1, n \rangle & \text{if } |\vec{x}| = n \\ \text{undefined} & \text{else} \end{cases} \end{aligned}$$

The definite terms are of the form  $f_1^n f_0$  or  $f_2 f_1^n f_0$ . The first grammar<sup>×</sup> is as follows.

$$(2.43) \quad \begin{aligned} \mathcal{J}_\varepsilon^\times(f_1)(\langle \vec{x}, n \rangle) &:= \vec{x}^\wedge 1 \\ \mathcal{J}_\varepsilon^\times(f_2)(\langle \vec{x}, n \rangle) &:= \begin{cases} \vec{x}^\wedge 1 & \text{if } |\vec{x}| = n \\ \text{undefined} & \text{else} \end{cases} \end{aligned}$$

$$(2.44) \quad \begin{aligned} \mathcal{J}_\mu^\times(f_1)(\langle \vec{x}, n \rangle) &:= n + 1 \\ \mathcal{J}_\mu^\times(f_2)(\langle \vec{x}, n \rangle) &:= n \end{aligned}$$

The second grammar<sup>×</sup> is as follows.

$$(2.45) \quad \begin{aligned} \mathcal{J}_\varepsilon^\times(f_1)(\langle \vec{x}, n \rangle) &:= \vec{x}^\wedge 1 \\ \mathcal{J}_\varepsilon^\times(f_2)(\langle \vec{x}, n \rangle) &:= \vec{x}^\wedge 1 \end{aligned}$$

$$(2.46) \quad \begin{aligned} \mathcal{J}_\mu^\times(f_1)(\langle \vec{x}, n \rangle) &:= n + 1 \\ \mathcal{J}_\mu^\times(f_2)(\langle \vec{x}, n \rangle) &:= \begin{cases} n & \text{if } |\vec{x}| = n \\ \text{undefined} & \text{else} \end{cases} \end{aligned}$$

The grammar  $G^\times$  is compositional but not autonomous; the grammar  $G^\times$  is autonomous but not compositional. The reason is this. In  $G^\times$  the functions  $\mathfrak{J}_\mu^\times(f_i)$  do not depend on the exponent, they are total and always yield a unique value. On the other hand,  $\mathfrak{J}_\varepsilon^\times(f_2)$  *does* depend on the meaning:

$$(2.47) \quad \mathcal{J}_\varepsilon^\times(f_2)(\langle 111, 2 \rangle) = \text{undefined}, \quad \mathcal{J}_\varepsilon^\times(f_2)(\langle 111, 3 \rangle) = 1111$$

Thus  $G^\times$  is indeed not autonomous but compositional. Likewise for the other claim. It might be deemed that it is possible to find a grammar<sup>×</sup> corresponding to  $G$  that is both autonomous and compositional. This is not possible. To see this, suppose  $G^\times = \langle \Omega, \mathcal{J}_\varepsilon^\times, \mathcal{J}_\mu^\times \rangle$  is such a grammar. Then for any given string  $\vec{x}$  there is some  $n$  (namely  $|\vec{x}|$ ) such that  $\mathcal{J}_\varepsilon^\times(f_2)(\langle \vec{x}, n \rangle)$  is defined. If the grammar is autonomous this means that for all  $m$   $\mathcal{J}_\varepsilon^\times(f_2)(\langle \vec{x}, m \rangle)$  is defined. Hence the function

$\mathcal{J}_\varepsilon^\omega(f_2)$  is total. Likewise we see that  $\mathcal{J}_\mu^\omega(f_2)$  is total. It follows that  $\text{dom}(\mathfrak{J}^\omega(f_2)) = \text{dom}(\mathcal{J}(f_2))$  is total. But this not the case. ♣

Given that we have shown categories to be eliminable it is worth knowing if the same applies for compositionality and autonomy. That is, we want to know whether if  $L$  is a compositional c-language also  $H[L]$  is. To define compositionality for c-languages, we simply need to stipulate that  $\mu(\langle e, c, m \rangle) := m$ , and  $\varepsilon(\langle e, c, m \rangle) := e$ , and then repeat Definition 2.5 almost verbatim. The following example now shows that compositionality and autonomy can be lost under reduction.

**Example 12.** Our example is based on the grammar of Example 11. We introduce a set  $C = \{d, o\}$  of categories. For any given triple  $\langle e, c, m \rangle$  we define

$$(2.48) \quad \begin{aligned} \mathcal{K}(f_1)(\langle e, c, m \rangle) &:= \begin{cases} \langle e^{-1}, d, m+1 \rangle & \text{if } c = d \\ \text{undefined} & \text{else} \end{cases} \\ \mathcal{K}(f_2)(\langle e, c, m \rangle) &:= \begin{cases} \langle e^{-1}, o, n \rangle & \text{if } c = d \\ \text{undefined} & \text{else} \end{cases} \end{aligned}$$

This grammar is such that all component functions are independent. Thus it is in particular independent. However, its reduction is not; it also neither autonomous (only extensionally autonomous) nor compositional (only extensionally compositional). For the reduction is exactly the grammar of Example 11.

Notice that the language generated by this grammar is independent. However, to generate it by an independent grammar we must choose a different signature. ♣

We close this section by some considerations concerning linguistic theories. First, the notion of a grammar as opposed to a grammar<sup>x</sup> has the drawback of not distinguishing between syntactically well-formed input and semantically well-formed input. Or, to phrase this in the technical language of this book, a term turns out to be semantically definite iff it is orthographically definite. It has a semantics iff it has an exponent. By switching to grammars<sup>x</sup> we create this possibility. However, as much as this might be desirable, it creates problems of its own. For now we have to decide which of the components is to blame for the fact that a term has no value. We can see to it that it is the syntax, or we can see to it that it is the semantics. If we add categories, there is a third possibility, namely to have a term whose category does not exist. Linguistic theories differ in the way they

handle the situation. Categorical Grammar is designed to be such that if a term is indefinite then it is categorially indefinite. That means, as long as a term has a category, it is also syntactically and semantically definite. This is *not* to say that there are no semantically indefinite terms. To the contrary, it was based on typed  $\lambda$ -calculus, so there were plenty of semantically ill-formed terms. But every time a term is semantically ill-formed it would automatically be categorially ill-formed. In LFG, each level has its own well-formedness conditions, so that one tries to explain the complexity of the output by factoring out which level is responsible for which output phenomenon. The theory is *modular*.

In generative grammar there was no separate level of categories. Technically, the syntax operated before semantics. Syntax operated autonomously from semantics. In the present formulation this just means that the syntactic functions do not respond to changes in the meaning (whence the name *autonomy* above). However, in our formulation there is no order in the way the terms are checked. The components of the sign are formed in parallel.

**Exercise 7.** Show how to generate the language of Example 11 using an independent grammar.

**Exercise 8.** Suppose that  $L$  is an interpreted language such that if  $\langle e, m \rangle, \langle e, m' \rangle \in L$  then  $m = m'$ . Show that  $L$  is extensionally autonomous.

**Exercise 9.** Suppose that  $L$  is an interpreted language such that if  $\langle e, m \rangle, \langle e', m \rangle \in L$  then  $e = e'$ . Show that  $L$  is extensionally compositional.

**Exercise 10.** Suppose that  $L \subseteq E \times M$  is an interpreted language which is a partial bijection between  $E$  and  $M$ . Then  $L$  is independent.

## 2.4 Leibniz' Principle

It may seem difficult to ascertain what exactly meanings are and how many we need. It is the purpose of this section to indicate that such worries are unfounded. Basically, we can—if certain conditions are met—assume that meanings are sets of exponents. For fix some  $m \in M$ . Given a grammar, there may or may not be an expression that has  $m$  as meaning. Now, assume first the following:

**Principle 4 (Effability)** *For any  $m \in M$  there is an exponent  $e$  such that  $\langle e, m \rangle \in L(G)$ .*

Given this, put

$$(2.49) \quad \{m\}_G := \{e : \langle e, m \rangle \in L(G)\}$$

Thus meanings are sets of exponents, those that exactly have the desired meaning.

One of the problems that one may have with (2.49) is that it does not really elucidate the matter. After all it relies on intuitions of identity in meaning, which often enough seems problematic. To replace that definition with something more tangible, we reduce the expressions that we consider to the set of *sentences*. The advantage of sentences is that they are either true or not. This requires a minimal amount of discriminatory ability to answer whether a given sentence is true or false. (I am ignoring quite a number of problems here that remain even if we only use propositions. Basically, I think that the reduction to sentences presents no advantages.) It seems—at least in principle—to get a unanimous answer to questions of the form: can sentences  $e_0, e_1, \dots, e_{n-1}$  together be true? Suppose then that we have answers of the form: “this or that set of sentences can be true”. Then we are able to identify the semantics of our language up to isomorphism, given that we assume Effability and Leibniz' Principle.

Leibniz' Principle is a principle that regulates how many meanings we must postulate. Its formulation requires some further apparatus. First we need to single out a special category of meanings, namely *propositions*. Propositions are such expressions of which we can say that they are *true*. Indeed, a **theory**  $T$  is said to be a set of propositions. Furthermore, a given proposition  $\varphi$  and a theory  $T$ , we say that  $\varphi$  is **true in**  $T$  if  $\varphi \in T$ . A **frame** is a set of theories. Given a frame  $\mathcal{F}$ , we say that  $\varphi$  and  $\chi$  are  $\mathcal{F}$ -equivalent if for all  $T \in \mathcal{F}$ :  $\varphi \in T$  iff  $\chi \in T$ . We write  $\varphi \approx_{\mathcal{F}} \chi$ . In ordinary terminology, we may think of theories as maximally consistent sets (or *worlds*) and of the frame as a Kripke-frame, though the relation will play no further role here. The frame is essential in spelling out what possibilities exist, which truths necessarily coexist. It embodies the meaning postulates. It does so via the Leibniz Principle. There may be requirements on the set of theories (which are usually codified as deducibility relations) but we shall not go into details here. What we do require, though, is the following version of Leibniz' Principle. A **polynomial** is a term in which variables have been replaced by constants denoting them. We assume that for every  $m$  there is a constant  $\underline{m}$  denoting it. If the language satisfies Effability, there always is a constant term denoting  $m$ , but we keep our definitions independent of that.

**Definition 2.8** *Suppose  $G$  is an interpreted grammar. Two meanings  $m$  and  $m'$*

are  $(G, \mathcal{F})$ -**equivalent** if whenever  $t(x)$  is a polynomial and  $t^\mu(m)$  is defined and a proposition, then also  $t^\mu(m')$  is defined and a proposition, and  $t^\mu(m) \approx_{\mathcal{F}} t^\mu(m')$ .

In this version, the Leibniz Principle ignores the exponents of signs and concentrates exclusively on the meanings. The function  $t^\mu(x)$  on the other hand is defined via the grammar.

**Proposition 2.9** *Suppose that for all  $i < \Omega(f)$ :  $m_i$  is  $(G, \mathcal{F})$ -equivalent to  $\hat{m}_i$ . Then  $\mathcal{J}(f)(m_0, \dots, m_{\Omega(f)-1})$  is  $(G, \mathcal{F})$ -equivalent to  $\mathcal{J}(f)(\hat{m}_0, \dots, \hat{m}_{\Omega(f)-1})$ .*

**Proof.** By contraposition. Suppose that  $\mathcal{J}(f)(m_0, \dots, m_{\Omega(f)-1})$  is not  $(G, \mathcal{F})$ -equivalent to  $\mathcal{J}(f)(\hat{m}_0, \dots, \hat{m}_{\Omega(f)-1})$ . Then there exists a term  $t$  such that either  $t^\mu(\mathcal{J}(f)(m_0, \dots, m_{\Omega(f)-1}))$  is a proposition but  $t^\mu(\mathcal{J}(f)(\hat{m}_0, \dots, \hat{m}_{\Omega(f)-1}))$  is not or conversely. It now follows that either  $m_0$  is not  $(G, \mathcal{F})$ -equivalent to  $\hat{m}_0$  (via  $s_0 := t(x, \underline{m_1}, \underline{m_2}, \dots, \underline{m_{\Omega(f)-1}}))$  or that  $m_1$  is not  $(G, \mathcal{F})$ -equivalent to  $\hat{m}_1$  (via  $s_0 := t(\underline{\hat{m}_0}, x, \underline{m_2}, \dots, \underline{m_{\Omega(f)-1}}))$  etc. In other words, for some  $i$ ,  $m_i$  is not  $(G, \mathcal{F})$ -equivalent with  $\hat{m}_i$ .  $\square$

This convoluted definition is necessary to be able to express both the notion of substitution (through the use of analysis terms) and truth equivalence (through the use of  $\mathcal{F}$ ).

**Principle 5 (Leibniz Principle)**  $m = m'$  iff  $m$  and  $m'$  are  $(G, \mathcal{F})$ -equivalent.

Given  $G$  and  $\mathcal{F}$ , let

$$(2.50) \quad \llbracket m \rrbracket := \{m' : m' \text{ (} G, \mathcal{F} \text{)-equivalent to } m\}$$

Assume that  $G$  is compositional. Then put

$$(2.51) \quad f^\#(\llbracket m_0 \rrbracket, \dots, \llbracket m_{\Omega(f)-1} \rrbracket) := \llbracket f_*^\mu(m_0, \dots, m_{\Omega(f)-1}) \rrbracket$$

By Proposition 2.9 this definition does not depend on the choice of representatives. Now replace  $f_*^\mu$  by  $f^\#$  in  $G$ . This gives the new grammar  $G^\#$ . This grammar satisfies Leibniz' Principle. Moreover, it is unique up to 'renaming the meanings'.

**Definition 2.10** *A  $G$ -congruence on  $M$  is an equivalence relation  $\sim$  such that if  $f$  is a mode and for all  $i < \Omega(f)$   $m_i \sim n_i$  then also  $f_*^\mu(\vec{m}) \sim f_*^\mu(\vec{n})$ .*

A congruence  $\sim$  defines a surjective map  $\kappa_{\sim} : m \mapsto \{m' : m' \sim m\}$ . This map is a compression in the sense of the following definition.

**Definition 2.11** A *compression* of  $G$  is an onto map  $\kappa : M \rightarrow N$  such that the relation  $\sim$  defined by  $m \sim n$  iff  $\kappa(m) = \kappa(n)$  is a  $G$ -congruence. Given a compression  $\kappa$ , we define  $G^{\kappa} = \langle \mathcal{J}^{\kappa}, \Omega \rangle$  over  $E \times N$  by

$$(2.52) \quad \begin{aligned} & \mathcal{J}^{\kappa}(f)(\langle e_0, \kappa(m_0) \rangle, \dots, \langle e_{\Omega(f)-1}, \kappa(m_{\Omega(f)-1}) \rangle) \\ & := \langle f^{\varepsilon}(\langle e_0, m_0 \rangle, \dots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1} \rangle), \\ & \quad \kappa(f^{\mu}(\langle e_0, m_0 \rangle, \dots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1} \rangle)) \rangle \end{aligned}$$

This definition is independent of representatives.

Clearly, if  $\kappa$  is a compression of  $G$ ,  $\sim_{\kappa}$  defined by  $m \sim_{\kappa} n$  iff  $\kappa(m) = \kappa(n)$  is a  $G$ -congruence. Moreover,  $\sim = \sim_{\kappa_{\sim}}$ .

A  $G$ -congruence  $\sim$  is  $\mathcal{F}$ -**compatible** if it follows that if  $m$  is  $(G, \mathcal{F})$ -equivalent to  $n$  then also  $m \sim n$ . Now, a grammar satisfies Leibniz Principle iff it has just one  $\mathcal{F}$ -compatible compression.

**Proposition 2.12** Any grammar  $G$  possesses a compression  $\kappa$  onto a grammar that satisfies the Leibniz Principle modulo  $\mathcal{F}$ . This grammar is unique in the following sense. For any two  $\mathcal{F}$ -compatible compression  $\lambda$  such that  $G^{\lambda}$  satisfies the Leibniz Principle there is a compatible compression  $\mu$  of  $G^{\kappa}$  such that  $(G^{\kappa})^{\mu} = G^{\lambda}$  and a compression  $\nu$  of  $G^{\lambda}$  onto  $G^{\kappa}$  such that  $\mu$  and  $\nu$  are inverses of each other.

**Exercise 11.** Suppose we drop Effability. Does the definition of  $\{m\}_G$  still make sense? Explain.



# Chapter 3

## Meanings

MEANINGS are the topic of this chapter. Unlike what is ordinarily assumed we do not consider the structure of the space of meanings and the functions on them a matter of arbitrary convention. Like we did with exponents, there is a genuine question of what meanings actually are and how they can be manipulated.

### 3.1 Desyntactified Meanings

Montague assumed that meanings are objects of a typed universe of functions. For this purpose we may either choose a universe of the typed  $\lambda$ -calculus or some version of typed combinatory logic. An essential tool is Currying, by which we can transform a function of several arguments into a function that takes the arguments one at a time. This allows to reduce the syntactic structure to binary branching. Additionally, it assumes that when two constituents are concatenated to form a new constituent, the meaning of the result is already determined, at least in the basic calculus. Namely, if two constituents can at all be put together into a single constituent then one of them will have type  $\alpha \rightarrow \beta$  and the other the type  $\alpha$ ; the result will therefore be of type  $\beta$ . The idea that constituent formation adds nothing to the meaning is also known as *lexicalism*. In this section I shall propose that rather than using functions we should use relations; and that we should also abandon lexicalism.

The present chapter is somewhat strange in view of the fact that we consider meaning a fixed input. Normally, it should not be possible to change the notion of semantics. However, the question of what meanings are is just as difficult as the question what syntactic representations are. While sentences appear at least on paper so that we can measure the correctness of what we say against some concrete facts, meanings are harder to pin down. Therefore, various different ideas about meanings have been put forward. For a long time the leading intuition was that meanings come in different flavours; each of the meanings have a different type to them. In this theory, sentences denoted propositions, which were completely decontextualised.

There is a standard procedure to eliminate functions from predicate logic. Likewise we shall show here that an approach based on functions can be replaced by one that uses relations only, or rather open propositions. An open proposition is of the form  $\varphi(x_0, x_1, \dots, x_{n-1})$  of type  $t$  ( $=$  truth value), where  $x_i$ ,  $i < n$  is a variable of any type. Thus, given an assignment of objects of appropriate type to the variables this expression will yield a truth value. Now, what I shall propose is that *every* constituent denotes an open proposition. This is rather easy to achieve. Suppose  $\vec{x}$  is an expression with meaning  $f$ . Then we replace the meaning  $f$  by  $x = f$ , where  $x$  is a variable of the same type as  $f$ . Now consider the rule of application:

$$(3.1) \quad A_{>}(\langle \vec{x}, M \rangle, \langle \vec{y}, N \rangle) = \langle \vec{x} \sqcap \vec{y}, M(N) \rangle$$

In the new semantics it becomes:

$$(3.2) \quad U_{>}(\langle \vec{x}, u = M \rangle, \langle \vec{y}, v = N \rangle) = \langle \vec{x} \sqcap \vec{y}, u = M \wedge v = N \wedge w = u(v) \rangle$$

This is however not always satisfactory. It introduces the idea of applying  $M$  to  $N$  through the construction; and the construction still speaks of applying  $M$  to  $N$ . There is an alternative, which runs as follows.

$$(3.3) \quad U_{>}(\langle \vec{x}, u = M(w) \rangle, \langle \vec{y}, v = N \rangle) = \langle \vec{x} \sqcap \vec{y}, u = M(w) \wedge v = N \wedge w = v \rangle$$

Now the rule simply conjoins the two meanings and unifies certain variables. If  $M(w)$  is a function and will have to be applied then we should also feed  $M(w)$  these additional arguments. In this way we can see to it that the generalised rule is as follows:

$$(3.4) \quad U_{>}^{ij}(\langle \vec{x}, \varphi(\vec{x}) \rangle, \langle \vec{y}, \chi(\vec{y}) \rangle) = \langle \vec{x} \sqcap \vec{y}, \varphi(\vec{x}) \wedge \chi(\vec{y}) \wedge x_i = y_j \rangle$$

Eliminating the equation we can alternatively write

$$(3.5) \quad U_{>}^{ij}(\langle \vec{x}, \varphi(\vec{x}) \rangle, \langle \vec{y}, \chi(\vec{y}) \rangle) = \langle \vec{x} \hat{\square} \vec{y}, \varphi(\vec{x}) \wedge [x_i/y_j]\chi(\vec{y}) \rangle$$

Thus we have following result: the meaning of a complex constituent is a conjunction of the meaning of its parts with some fixed open formula. This is a welcome result. For it says that every meaning is propositional, and merging two constituents is conjunction—up to the addition of some more constraints.

## 3.2 The Space of Meanings

We assume that basic objects are sortal; we have, for example, *objects*, *time points*, *regions*, *worlds*, *truth values*, and so on. For each of these sorts we assume that the meanings associated with it come from a particular set. Thus we assume that we have a primitive set  $S$  sorts together with a family of sets  $\{M_s : s \in S\}$ . A **type** is a member of  $S^*$ , that is, it is a string of sorts. For any type  $\vec{s}$ , an **object** of type  $\vec{s}$  is an element of the set  $M_{\vec{s}}$ , which is defined inductively as follows.

$$(3.6) \quad \begin{aligned} M_{\langle \rangle} &:= \{\emptyset\} \\ M_{\langle s \rangle} &:= M_s \\ M_{\vec{s};t} &:= M_{\vec{s}} \times M_t \end{aligned}$$

Finally, a **relation of type**  $\vec{s}$  is a set of objects of type  $\vec{s}$ . The type  $\langle \rangle$  is special importance. It corresponds to the set  $\{\emptyset\}$ . This set has two subsets:  $0 := \emptyset$  and  $1 := \{\emptyset\}$ . These sets will function as truth values.

As it will turn out, we are not interested in relations but in a more abstract notion, that of a *concept*. A concept is a set of relations which are in some sense variants of each other. A relation is a variant of another relation if it can be obtained either by permutation of its arguments or by contracting or expanding it. A precise definition is as follows.

Let  $\vec{s} = \langle s_0, s_1, \dots, s_{n-1} \rangle$  be a type and  $\pi : \{0, 1, \dots, n-1\} \rightarrow \{0, 1, \dots, n-1\}$  be a permutation. Then  $\pi(\vec{s}) := \langle s_{\pi(0)}, s_{\pi(1)}, \dots, s_{\pi(n-1)} \rangle$  is a **permutation** of  $\vec{s}$ . If  $t \in S$  then  $\vec{s};t$  is an **expansion** of  $\vec{s}$ . Given a relation  $R$  of type  $\vec{s}$ , define

$$(3.7) \quad \pi[R] := \{ \langle x_{\pi(0)}, x_{\pi(1)}, \dots, x_{\pi(n-1)} \rangle : \langle x_0, x_1, \dots, x_{n-1} \rangle \in R \}$$

This is a relation of type  $\pi(\vec{s})$ . A relation  $R'$  is said to be a **permutation** of  $R$  iff it is of the form  $\pi[R]$  for some permutation  $\pi$ . Furthermore, let

$$(3.8) \quad E[R] := \{\langle x_0, x_1, \dots, x_{n-1}, x_{n-1} \rangle : \langle x_0, x_1, \dots, x_{n-1} \rangle \in R\}$$

This is a relation of type  $\vec{s}, s_{n-1}$ . A relation  $R'$  is said to be a **diagonal expansion** of  $R$  iff it has the form  $E[R]$ . Finally, set

$$(3.9) \quad P_t[R] := \{\langle x_0, x_1, \dots, x_{n-1}, x_n \rangle : \langle x_0, x_1, \dots, x_{n-1} \rangle \in R, x_n \in M_t\}$$

This is a relation of type  $t$ . A relation is said to be a **product expansion of  $R$**  (with type  $t$ ) iff it has the form  $P_t[R]$ .

**Definition 3.1**  $R'$  is an **immediate variant** of  $R$  iff  $R'$  is either a permutation of  $R$  or  $R'$  is a diagonal expansion of  $R$  or  $R$  is a diagonal expansion of  $R'$  or  $R'$  is a product expansion of  $R$  or  $R$  is a product expansion of  $R'$ .  $R'$  is a **variant** of  $R$  if there is a series  $\langle R_i : i < n \rangle$  such that  $R_0 = R$ ,  $R_{n-1} = R'$  and for each  $i < n - 1$ ,  $R_{i+1}$  is an immediate variant of  $R_i$ . We write  $R \sim R'$  if  $R'$  is a variant of  $R$ .

**Example 13.** Let  $S = \{\ell, n\}$ ,  $M_\ell = \{a, b, c\}$  and  $M_n = \{0, 1\}$ . The relation  $R = \{\langle a, 0 \rangle, \langle b, 1 \rangle\}$  is of type  $\langle \ell, c \rangle$ . It has a nonidentical permutation  $R' = \{\langle 0, a \rangle, \langle 1, b \rangle\}$ . This is also known as the **converse** of  $R$ , and written  $R^\sim$ . The diagonal expansion of  $R$  is  $E[R] := \{\langle a, 0, 0 \rangle, \langle b, 1, 1 \rangle\}$ . The diagonal expansion of  $R'$  is  $E[R'] = \{\langle 0, a, a \rangle, \langle 1, b, b \rangle\}$ . Thus, even though the diagonal expansion repeats only the last column,  $R$  has many more variants. Write

$$(3.10) \quad E_i[R] := \{\langle x_0, x_1, \dots, x_{n-1}, x_i \rangle : \langle x_0, x_1, \dots, x_{n-1} \rangle \in R\}$$

Then  $E_i[R]$  is a variant of  $R$ . Namely, let  $\pi = (i \ n - 1)$  and  $\pi' = (i \ n)$ . These are the permutations that exchange the items number  $i$  and  $n - 1$  in the case of  $\pi$ , and  $i$  and  $n$  in the case of  $\pi'$ . Then

$$(3.11) \quad E_i[R] = \pi'[E[\pi[R]]]$$

We say that  $R'$  is a **generalised diagonal expansion** of  $R$  if  $R' = E_i[R]$  for some  $i$ . Likewise, the generalised product expansion is defined by

$$(3.12) \quad P_t^i[R] := \{\langle x_0, x_1, \dots, x_{n-1}, x_n \rangle : \langle x_0, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_{n-1} \rangle \in R, x_i \in M_t\}$$



Notice the following. The identity relation of type  $\langle s, s \rangle$  is defined as

$$(3.13) \quad \{\langle x, x \rangle : x \in M_s\}$$

This is a product expansion of type  $s$  of the total relation  $M_s$  of type  $\langle s \rangle$ . This in turn is a product expansion of the relation  $M_{\langle \rangle} = \{\emptyset\} = 1$ . Thus the identity relation is a variant of the ‘true’ relation. This will consequences to be considered in more detail later.

**Definition 3.2** A *concept* is a set of relations of the form  $\llbracket R \rrbracket := \{R' : R' \sim R\}$ .

We shall now describe an interpretation of predicate logic. A formula is interpreted as a set of assignments. The language  $L_\tau$  consists of the following:

1. variables  $x_i^s$ , where  $i \in \mathbb{N}$ , and  $s \in S$ , the set of types;
2. relation symbols  $R$  of type  $\tau(R)$ ;
3. propositional connectives  $\wedge, \vee, \neg$ ;
4. for each  $i \in \mathbb{N}$ , and type  $s \in S$ , quantifiers  $\exists x_i^s$  and  $\forall x_i^s$ .

For each  $s \in S$  there is an identity  $=^s$ , which we normally write  $=$ . A  $\tau$ -**structure** is a pair  $\mathcal{M} = \langle \mathcal{M}, \mathcal{J} \rangle$ , where  $\mathcal{M} = \{M_s : s \in S\}$  and for every relation symbol  $R$ ,  $\mathcal{J}(R)$  is a relation of type  $\tau(R)$  over  $\mathcal{M}$ . An **assignment** into  $\mathcal{M}$  is defined as a function  $\beta$  from the set of variables into  $\bigcup \mathcal{M} := \bigcup_{s \in S} M_s$  such that for every  $s \in S$ :  $\beta(x_i^s) \in M_s$ . Ordinarily, a formula  $\varphi(x_0, x_1, \dots, x_{n-1})$  with variables  $x_i$  of type  $s_i$  is interpreted as a relation of type  $\vec{s} := \langle s_0, s_1, \dots, s_{n-1} \rangle$ . We shall take a detour via the assignments. Write  $[\varphi]_{\mathcal{M}}$  for the set of assignments making  $\varphi$  true. It is defined inductively. For a given assignment  $\beta$ , write  $\beta' \sim_{x_i^s} \beta$  if for all  $t \neq s$  and all  $j \neq i$ :  $\beta'(x_j^t) = \beta(x_j^t)$ .  $V$  is the set of all assignments.

$$(3.14) \quad \begin{aligned} [R(\vec{y})]_{\mathcal{M}} &:= \{\beta : \langle \beta(y_0), \beta(y_1), \dots, \beta(y_{n-1}) \rangle \in \mathcal{J}(R)\} \\ [\neg \varphi]_{\mathcal{M}} &:= V - [\varphi]_{\mathcal{M}} \\ [\varphi \wedge \chi]_{\mathcal{M}} &:= [\varphi]_{\mathcal{M}} \cap [\chi]_{\mathcal{M}} \\ [\varphi \vee \chi]_{\mathcal{M}} &:= [\varphi]_{\mathcal{M}} \cup [\chi]_{\mathcal{M}} \\ [\exists x_i^s \varphi]_{\mathcal{M}} &:= \{\beta : \text{there is } \beta' \sim_{x_i^s} \beta : \beta' \in [\varphi]_{\mathcal{M}}\} \end{aligned}$$

What we really want to do, though, is to interpret formulae not by sets of assignments but by concepts. The definition is as follows. Let  $\text{fr}(\varphi)$  denote the set of free formulae. Assume that  $\text{fr}(\varphi) = \{y_0, y_1, \dots, y_{n-1}\}$ . Then

$$(3.15) \quad \llbracket \varphi \rrbracket_{\mathcal{M}} := \{ \vec{a} : \text{there is a } \beta \in [\varphi]_{\mathcal{M}} : \text{for all } i < n : \beta(y_i) = a_i \}$$

This means the following. The concept denoted by the formula  $\varphi$  is that concept that consist of all the variants of the set of  $\langle a_0, a_1, \dots, a_{n-1} \rangle$  that satisfy the formula  $\varphi(y_0, y_1, \dots, y_{n-1})$ . Let us see some consequences of this definition. First, assume that  $x_k^s$  does not occur freely in  $\varphi(\vec{y})$ . Then for the formula  $\chi := \varphi(\vec{y}) \wedge x_k^s = x_k^s$  we get  $\text{fr}(\chi) = \text{fr}(\varphi) \cup \{x_k^s\}$ . Furthermore, we have

$$(3.16) \quad [\varphi]_{\mathcal{M}} = [\chi]_{\mathcal{M}}$$

since both formulae are satisfied by the same assignments. Nevertheless, the relations to be computed are different. The first is the set

$$(3.17) \quad K = \{ \langle \beta(y_0), \beta(y_1), \dots, \beta(y_{n-1}) \rangle : \beta \in [\varphi]_{\mathcal{M}} \}$$

The second is the set

$$(3.18) \quad L = \{ \langle \beta(y_0), \beta(y_1), \dots, \beta(y_{n-1}), \beta(x_k^s) \rangle : \beta \in [\varphi]_{\mathcal{M}} \} = K \times M_s$$

Since  $K \times M_s$  is a product expansion of  $K$ , however, we have  $\llbracket \chi \rrbracket_{\mathcal{M}} = \llbracket \varphi \rrbracket_{\mathcal{M}}$ . Thus the addition of ‘trivial’ variables has no effect on the concept. But there is more. Notice, for example, that the concept does not depend on the way we number the  $y_i$ . The relation will be a permutation of the original relation, which by definition is a variant of it. Additionally, let  $\chi(y_1, y_0) := \varphi(y_0, y_1)$ . Then  $\llbracket \chi \rrbracket_{\mathcal{M}} = \llbracket \varphi \rrbracket_{\mathcal{M}}$ . It is therefore the case that

$$(3.19) \quad \llbracket x_0^e < x_1^e \rrbracket_{\mathcal{M}} = \llbracket x_0^e > x_1^e \rrbracket_{\mathcal{M}}$$

In other words, for objects of sort  $e$  the concept of ‘being smaller than’ is the same concept as ‘being bigger than’. This looks like a contradiction, but it is not. The idea is that although the concept contains both relations, in the formation of complex formulae just one of them is being used at a time. This is achieved by the so-called *linking aspect*, to which we now turn.

### 3.3 Linking Aspects

Let us now see if we can define  $\langle\varphi\rangle_{\mathcal{M}}$  by induction on  $\varphi$ . The base case is straightforward. However, conjunction already presents problems. Consider the formulae  $\chi_0 := \langle x_0 < x_1 \wedge x_1 < x_0 \rangle_{\mathcal{M}}$  and  $\chi_1 := \langle x_0 < x_1 \wedge x_0 < x_1 \rangle_{\mathcal{M}}$ . (We shall sort when they are not relevant.) The parts from which the formulae are made, are  $x_0 < x_1$  and  $x_1 < x_0$ , both however denote the same concept! Thus, if there is a single function on concepts, say,  $\odot$ , which forms the intersection we get the paradoxical result

$$\begin{aligned}
 \langle\chi_0\rangle_{\mathcal{M}} &= \langle x_0 < x_1 \rangle_{\mathcal{M}} \odot \langle x_1 < x_0 \rangle_{\mathcal{M}} \\
 (3.20) \quad &= \langle x_0 < x_1 \rangle \odot \langle x_0 < x_1 \rangle_{\mathcal{M}} \\
 &= \langle\chi_1\rangle_{\mathcal{M}} \\
 &= \llbracket \emptyset \rrbracket
 \end{aligned}$$

What has gone wrong? Recall that a concept is a set of relations; if we want to intersect two concepts  $\mathfrak{c}$  and  $\mathfrak{d}$ , the result will be the concept corresponding to the intersection of two sets, say  $P \in \mathfrak{c}$  and  $Q \in \mathfrak{d}$ . However, what concept we get depends on the choice of  $P$  and  $Q$ . In the present case we have  $P \cap P^{\sim} = \emptyset$ . Since  $P^{\sim} \in \llbracket P \rrbracket$ , the intersection may be—among other—the concept  $\llbracket P \cap P^{\sim} \rrbracket = \llbracket P \rrbracket$  or the concept  $\llbracket P \cap P^{\sim} \rrbracket = \llbracket \emptyset \rrbracket$ .

To remedy the situation we have to choose a good representative of the concept. Which one we choose will in the present case depend on the formulae that represent the concepts. The details are as follows.

**Definition 3.3** *The **length** of a relation  $R$  is the length of any member of  $R$ . Let  $\mathfrak{c}$  be a concept. A relation  $R \in \mathfrak{c}$  is **minimal** in  $\mathfrak{c}$  if it is of minimal length among all members of  $\mathfrak{c}$ .*

Minimal relations obviously exist; moreover, they are in an important sense unique.

**Theorem 3.4** *Let  $R$  and  $R'$  be minimal members of a concept  $\mathfrak{c}$ . Then  $R$  is a permutation of  $R'$ .*

**Proof.** It follows from the definition that if  $R$  and  $R'$  are minimal, then they are of equal length. Let  $R$  be a minimal relation of length  $n$ . By induction, we define

a map  $h : \{0, 1, \dots, n-1\} \rightarrow \{0, 1, \dots, k-1\}$  for any variant  $R'$  of  $R$ . This map has the following property: for any given tuple  $\vec{a}$  of length  $k$ , put  $h(\vec{a}) = \langle a_{h(0)}, a_{h(1)}, \dots, a_{h(n-1)} \rangle$ , then  $(*) h[R']$  is a permutation of  $R$ . This map is therefore injective, and if  $h$  is bijective then  $R'$  is a permutation of  $R$ . We begin with the variant  $R$ ; the map is the identity.  $h$  has the property  $(*)$ . Let  $R'$  be a variant, and the map be  $h'$ . Now let  $R''$  be an expansion of  $R'$ . Then  $h'' := h'$ . If  $R'' = \pi[R']$ , then put  $h''(i) := \pi(h'(i))$ . Suppose,  $R''$  is a product expansion; then  $R'' = R' \times M_s$  for some  $s \in S$ . And

$$\begin{aligned}
 (3.21) \quad h''[R''] &= \{\langle a_{h''(0)}, a_{h''(1)}, \dots, a_{h''(n-1)} \rangle : \vec{a} \in R''\} \\
 &= \{\langle a_{h'(0)}, a_{h'(1)}, \dots, a_{h'(n-1)} \rangle : \vec{a} \in R''\} \\
 &= \{\langle a_{h'(0)}, a_{h'(1)}, \dots, a_{h'(n-1)} \rangle : \vec{a} \in R'\} \\
 &= h'[R']
 \end{aligned}$$

The step from the second to the third line is valid because the last element of  $\vec{a} \in R''$  is never used, so we might as well drop it. By inductive hypothesis,  $h'[R']$  is a permutation of  $R$ ; so is therefore  $h''[R'']$ . Similarly if  $R''$  is a diagonal expansion of  $R'$ .

Finally, assume that  $R'$  is a diagonal expansion (or a product expansion) of  $R''$ . Two cases arise. (a)  $k-1$  is not in the image of  $h'$ ; then we define  $h'' := h'$ . The verification of  $(*)$  now proceeds as before. (b)  $k-1 = h'(i)$  for some  $i < n$ . Then we put  $h''(i) := k-2$ . Furthermore, for all  $j \neq i$ ,  $h''(j) := h'(j)$ . First of all,  $k-2$  is not in the image of  $h'$ . For suppose the contrary, and that  $k-2 = h'(j)$ . Then it follows, by definition of a diagonal expansion, that for all  $\vec{c} = h'[R']$ :  $\vec{c}$  has the form  $\langle a_{h'(0)}, a_{h'(1)}, \dots, a_{h'(n-1)} \rangle$  for some  $\vec{a} \in R'$ , where  $a_{h'(j)} = a_{h'(i)}$ . Since  $h'(j) \neq h'(i)$ , and since  $h'[R']$  is a permutation of  $R$ , it follows that  $R$  is not minimal: for we can contract the last columns  $i$  and  $j$ . Thus,  $h''$  is well-defined.  $(*)$  is shown as follows.

$$\begin{aligned}
 (3.22) \quad h''[R''] &= \{\langle a_{h''(0)}, a_{h''(1)}, \dots, a_{h''(n-1)} \rangle : \vec{a} \in R''\} \\
 &= \{\langle a_{h'(0)}, a_{h'(1)}, \dots, a_{h'(n-1)} \rangle : \vec{a} \in R''\} \\
 &= \{\langle a_{h'(0)}, a_{h'(1)}, \dots, a_{h'(n-1)} \rangle : \vec{a} \in R'\} \\
 &= h'[R']
 \end{aligned}$$

The step from the first to the second line is valid since for all  $p < n-1$ ,  $a_{h''(p)} = a_{h'(p)}$ ; for either  $p \neq i$  and then  $h''(p) = h'(p)$ , or  $p = i$ , and then  $a_{k-1} = a_{h''(p)} = a_{h'(p)} = a_{k-1}$ , by assumption (b). The remaining case is left to the reader.  $\square$

The proof reveals that the concept allows to define the generating relation up to a permutation on condition that the generating relation is nonreducible, that is, cannot be obtained from another relation by expansion.

**Definition 3.5** *A linking aspect is a function  $Y$  defined on the set of concepts such that  $Y(c)$  is a minimal member of  $c$ .*

We can define a function  $\otimes^Y$  on the basis of a linking aspect by

$$(3.23) \quad c \otimes^Y d := \llbracket Y(c) \wedge Y(d) \rrbracket$$

The right hand side need not be defined, which is the case exactly if the types of the sets  $Y(c)$  and  $Y(d)$  do not match. The solution to the above paradox will therefore consist in choosing the correct linking aspect. Notice, though, that it will require a different formulation, where the linking aspect is chosen on the basis of the actual names given to the variables. This means that there is no compositional interpretation of the language at hand. We shall argue below that this is a welcome consequence.

A more general conjunction is the following; let  $Y$  and  $Z$  be two linking aspects. Define

$$(3.24) \quad c \otimes^{YZ} d := \llbracket Y(c) \wedge Z(d) \rrbracket$$



# Chapter 4

## Examples

FINALLY we shall present some examples. The first example will be standard predicate logic.

### 4.1 Predicate Logic

In this section we present the standard predicate logic. There will be no sorts; they do not add anything of significance. The alphabet is the following set:  $A := \{ (, ), , , 0, 1, x, \rightarrow, \neg, \vee, \wedge, \exists, \forall \} \cup \Delta \cup \Theta$ , where  $\Delta$  contains the relation symbols and  $\Theta$  the function symbols. The two sets are assumed to be disjoint. The arity of  $\delta \in \Delta \cup \Theta$  is given by  $a(\delta)$ . We shall first describe informally the formation rules of well-formed expressions and their denotation and then present a grammar of the interpreted language. The interpretation will be given in terms of a structure  $\mathcal{M} = \langle M, \mathfrak{I} \rangle$ , where  $M$  is a set and  $\mathfrak{I}$  a function sending a relation symbol  $R$  to a set  $\mathfrak{I}(R) \subseteq M^{a(R)}$  and a function symbol  $f$  to a function  $\mathfrak{I}(f) : M^{a(f)} \rightarrow M$ . A **valuation** is a function  $\beta : \{0, 1\} \rightarrow M$ . The set of all valuations is denoted by  $V$ .

An **index** is a sequence of 0 and 1. The denotation of an index is the index itself. A **variable** is a sequence  $x \vec{y}$ , where  $\vec{y}$  is an index. The denotation of the variable is the function  $\vec{y}^* : \beta \mapsto \beta(\vec{y})$ . A **term** is either a variable or an expression of the form  $f(\vec{v}_0, \vec{v}_1 \cdots \vec{v}_{a(f)-1})$ . Its meaning is the function  $m(f) : \beta \mapsto \mathfrak{I}(f)(m(\vec{v}_0)(\beta), m(\vec{v}_1)(\beta), \cdots, m(\vec{v}_{a(f)-1})(\beta))$ . An **atomic formula** is an expression of the form  $R(\vec{v}_0, \vec{v}_1 \cdots \vec{v}_{a(R)-1})$ , where the  $\vec{v}_i$  are terms. Its meaning is

the set  $m(R) := \{\beta : \langle m(\vec{v}_0)(\beta), m(\vec{v}_1)(\beta), \dots, m(\vec{v}_{a(R)-1})(\beta) \rangle \in \mathfrak{I}(R)\}$ . Complex formulae are of the form  $(\neg\varphi)$ ,  $(\varphi\wedge\chi)$ ,  $(\varphi\vee\chi)$ ,  $(\varphi\rightarrow\chi)$ ,  $(\exists\mathbf{x}\vec{v})\varphi$ ,  $(\forall\mathbf{x}\vec{v})\varphi$ , where  $\vec{v}$  is an index,  $\varphi$  and  $\chi$  are formulae. Their meaning has been spelled out earlier in Section 3.2.

This finishes the definition of  $L_{\Delta, \mathcal{M}}$ . Now we shall present a grammar for that language. We shall use the following modes:

$$(4.1) \quad F := \{f_\emptyset, f_0, f_1, f_v, f_\neg, f_\wedge, f_\vee, f_\rightarrow, f_\exists, f_\forall\} \cup \{f_\delta : \delta \in \Delta \cup \Theta\}$$

The signature is  $\Omega : f_\emptyset \mapsto 0, f_1 \mapsto 1, f_2 \mapsto 1, f_v \mapsto 1, f_\neg \mapsto 1, f_\wedge \mapsto 2, f_\vee \mapsto 2, f_\rightarrow \mapsto 2, f_\exists \mapsto 2, f_\forall \mapsto 2, f_\delta \mapsto a(\delta)$ , where  $\delta \in \Delta \cup \Theta$ . First, we shall define the modes that build up variables:

$$(4.2) \quad \begin{aligned} \mathcal{C}(f_\emptyset) &:= \langle \varepsilon, \varepsilon \rangle \\ \mathcal{C}(f_0)(\langle e, m \rangle) &:= \begin{cases} \langle e^{\frown} 0, m^{\frown} 0 \rangle & \text{provided that } e \text{ is an index} \\ \text{undefined} & \text{else} \end{cases} \\ \mathcal{C}(f_1)(\langle e, m \rangle) &:= \begin{cases} \langle e^{\frown} 1, m^{\frown} 1 \rangle & \text{provided that } e \text{ is an index} \\ \text{undefined} & \text{else} \end{cases} \\ \mathcal{C}(f_v)(\langle e, m \rangle) &:= \begin{cases} \langle \mathbf{x}^{\frown} e, m^* \rangle & \text{provided that } e \text{ is an index} \\ \text{undefined} & \text{else} \end{cases} \end{aligned}$$

Recall that  $e^*(\beta) = \beta(e)$ , the function that is defined on assignments and applies the assignment to the index. Next we turn to functions and relations. Let  $g$  be a function. The corresponding mode is.

$$(4.3) \quad \begin{aligned} \mathcal{C}(f_g)(\langle e_0, m_0 \rangle, \dots, \langle e_{a(g)-1}, m_{a(g)-1} \rangle) \\ := \begin{cases} \langle g^{\frown} (e_0^{\frown}, \dots, e_{a(g)-1}^{\frown}), \lambda\beta. \mathcal{J}(g)(m_0(\beta), \dots, m_{a(g)-1}(\beta)) \rangle \\ \text{if the } e_i \text{ are terms} \\ \text{undefined else} \end{cases} \end{aligned}$$

Let  $R$  be a relation:

$$(4.4) \quad \begin{aligned} \mathcal{C}(f_R)(\langle e_0, m_0 \rangle, \dots, \langle e_{a(R)-1}, m_{a(R)-1} \rangle) \\ := \begin{cases} \langle R^{\frown} (e_0^{\frown}, \dots, e_{a(R)-1}^{\frown}), \{\beta : \langle m_0(\beta), \dots, m_{a(R)-1}(\beta) \rangle \in \mathcal{J}(R)\} \rangle \\ \text{if the } e_i \text{ are terms} \\ \text{undefined else} \end{cases} \end{aligned}$$

Finally we introduce the modes for the connectives. No difficulties arise with the booleans:

$$(4.5) \quad \begin{aligned} \mathcal{C}(f_{\neg})(\langle e, m \rangle) &:= \begin{cases} \langle (\neg \neg e), V - m \rangle & \text{if } e \text{ is a formula} \\ \text{undefined} & \text{else} \end{cases} \\ \mathcal{C}(f_{\wedge})(\langle e_0, m_0 \rangle, \langle e_1, m_1 \rangle) &:= \begin{cases} \langle (\neg e_0 \wedge \neg e_1), m_0 \cap m_1 \rangle \\ \text{if } e_0 \text{ and } e_1 \text{ are formulae} \\ \text{undefined} & \text{else} \end{cases} \\ \mathcal{C}(f_{\vee})(\langle e_0, m_0 \rangle, \langle e_1, m_1 \rangle) &:= \begin{cases} \langle (\neg e_0 \vee \neg e_1), m_0 \cup m_1 \rangle \\ \text{if } e_0 \text{ and } e_1 \text{ are formulae} \\ \text{undefined} & \text{else} \end{cases} \\ \mathcal{C}(f_{\rightarrow})(\langle e_0, m_0 \rangle, \langle e_1, m_1 \rangle) &:= \begin{cases} \langle (\neg e_0 \rightarrow \neg e_1), (V - m_0) \cup m_1 \rangle \\ \text{if } e_0 \text{ and } e_1 \text{ are formulae} \\ \text{undefined} & \text{else} \end{cases} \end{aligned}$$

Finally the quantifiers. They are introduced by binary modes, one responsible for the handling of the variable and the other responsible for the scope. The definition is somewhat tricky. We assume that  $M$  has at least two elements, say  $a$  and  $b$ . Given an index  $\vec{y}$ , let  $\beta_a^{\vec{y}}$  be the valuation that assigns  $a$  to  $\vec{y}$  and  $b$  to every other variable. If  $m$  has the form  $v^*$  for some variable  $v$  then we can find the index of that variable by looking at the unique  $\vec{y}$  such that  $\vec{y}^*(\beta_a^{\vec{y}}) = a$ . We denote the variable with index  $\vec{y}$  by  $v(m)$ .

$$(4.6) \quad \begin{aligned} \mathcal{C}(f_{\exists})(\langle e_0, m_0 \rangle, \langle e_1, m_1 \rangle) &:= \begin{cases} \langle (\neg \exists \neg e_0) \wedge e_1, \{\beta' : \text{exists } \beta \sim_{v(m_0)} \beta' : \beta \in m_1\} \rangle \\ \text{if } e_0 \text{ is a variable and } e_1 \text{ a formula} \\ \text{undefined} & \text{else} \end{cases} \end{aligned}$$

If  $M$  contains just one element then we put

$$(4.7) \quad \mathcal{C}(f_{\exists})(\langle e_0, m_0 \rangle, \langle e_1, m_1 \rangle) := \begin{cases} \langle (\neg \exists \neg e_0) \wedge e_1, m_1 \rangle \\ \text{if } e_0 \text{ is a variable and } e_1 \text{ a formula} \\ \text{undefined} & \text{else} \end{cases}$$

The universal quantifier is quite similar. This finishes the definition of the grammar. Let us notice that this grammar is actually independent. The functions on the

exponents and the functions on the meanings are independently formulated. In this case what needs to be checked is that the domains for these functions (which are partial) are independently specifiable. As we have spelled out the grammar, the functions on the exponents are partial, and the conditions on the mode are spelled out as conditions on the exponents. Hence this is unproblematic. Now, the functions on the meaning are *de facto* partial. Yet in case the functions on the exponents are defined, the meanings can also be composed, and therefore no supplementary condition needs to be added.

I should mention here that [Fine, 2003] has claimed that there is no compositional semantics for predicate logic. The above grammar suggests that this is false. Indeed, what Fine has in mind is a different interpretation of predicate logic by which we do not use variables that consist of, say, a letter and an index. Rather, he has in mind a semantics where the name of the variable is immaterial; this corresponds to the factual *use* of predicate logic in everyday discourse, even in logic. Careful texts admit that what they are using are not actual variables but metavariables. If we want to give a semantics of predicate logic in terms of metavariables we must change the definitions rather substantially. This is what we shall do next.

## 4.2 Concept Based Predicate Logic

In this section we shall explore the kinds of operations that grammars may use. Ideally, one would expect that we can define a grammar for the language of concepts over the set  $\Delta$  of relations and  $\Theta$  of functions (whether sorted or not is of little importance). This is the following language:

$$(4.8) \quad \text{CL}_{\Delta \cup \Theta} := \{ \langle \varphi, \langle \varphi \rangle_{\mathcal{M}} \rangle : \varphi \in \text{PL}_{\Delta \cup \Theta} \}$$

There is a trivial sense in which this is possible: what we need to do is use the formation rules of the previous section and define the meaning functions  $f^\mu$  simply by

$$(4.9) \quad f^\mu(\langle e_0, m_0 \rangle, \dots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1} \rangle) := \langle f_*^e(\vec{e}), \langle f_*^e(\vec{e}) \rangle_{\mathcal{M}} \rangle$$

In plain words: we first form the exponent (which we can do since the grammar of the previous section is autonomous) and then simply take as the meaning the concept defined by the exponent. The problem is that this grammar is not compositional. The question therefore is whether we can give a compositional grammar for the language of concepts.

It will turn out to be impossible to do this. There are too many complications that arise. Some of them will be illustrated below. However, principally we are interested in the semantics of natural languages, and so the negative results is not really a disappointment. For it may still turn out that languages have a compositional grammar; it just so happens that they are expressively different from predicate logic.

The problems that we face are as follows:

1. There are unboundedly many free variables in the formulae.
2. The linking aspects cannot always be sufficiently defined.

It will turn out that only the first presents an insurmountable difficulty. Once we restrict ourselves to finite variable fragments, compositional grammars can be given.

Let us review these problems in turn. Consider the formula (in informal notation)

$$(4.10) \quad \varphi_n(x_0, \dots, x_n) := x_0 < x_1 \wedge x_1 < x_2 \wedge \dots \wedge x_{n-1} < x_n$$

Now consider the conjunction

$$(4.11) \quad \varphi_n(\vec{x}) \wedge \varphi_n(x_k, x_{k+1}, \dots, x_{k+n})$$

This formula is obtained from  $\varphi_n(\vec{x})$  and  $\varphi_n(x_k, \dots, x_{k+n})$ . Its value depends on  $k$ . Observe however that

$$(4.12) \quad \llbracket \varphi(x_0, \dots, x_n) \rrbracket_{\mathcal{M}} = \llbracket \varphi(x_k, \dots, x_{k+n}) \rrbracket_{\mathcal{M}}$$

Observe also that for at least  $n + 2$  different choices (from 0 to  $n + 1$ ) the concepts

$$(4.13) \quad \llbracket \varphi_n(\vec{x}) \wedge \varphi_n(x_k, x_{k+1}, \dots, x_{k+n}) \rrbracket_{\mathcal{M}}$$

are all distinct. This means that if we use the functions on exponents of the previous grammar, there is no way to construct all necessary formulae, since we need an unbounded array of modes. This is of course no proof that no grammar exists; however, to perform such a proof all that is needed is a series of independent formulae in the variables  $x_0$  through  $x_{n-1}$  for every  $n$ . They can be constructed.

Let us therefore assume that we use only the formulae with up to  $n$  free variables, for some  $n$ . It is not necessary that they are called  $x_0$  through  $x_{n-1}$ . However, to keep matters simple we shall remain with the language  $\text{PL}_\Delta^n$ , which is the fragment of predicate logic with relations in  $\Delta$  and variables from  $\Theta$ . Functions will also be omitted. Now fix a structure  $\mathcal{M} = \langle M, \mathcal{J} \rangle$ . We put

$$(4.14) \quad \llbracket \varphi \rrbracket_{\mathcal{M}} := \llbracket [\varphi]_{\mathcal{M}} \rrbracket$$

We shall present an independent grammar for

$$(4.15) \quad L_n = \{ \langle \varphi, \llbracket \varphi \rrbracket_{\mathcal{M}} \rangle : \varphi \in \text{PL}_\Delta \}$$

Define  $C := \{ \llbracket \varphi \rrbracket_{\mathcal{M}} : \varphi \in L_n \}$ , the set of  $L_n$ -definable concepts in  $\mathcal{M}$ . Let  $f : C \rightarrow L_n$  be a function such that  $c = \llbracket f(c) \rrbracket_{\mathcal{M}}$ . Thus,  $f$  picks for each concept a formula defining it. With respect to this formula we assign to a formula  $\chi$  a **type** in the following way. Let  $\Pi_n$  be the set of permutations of  $\{0, 1, \dots, n-1\}$ . Then the type of  $\chi$  is that permutation  $\pi \in \Pi_n$  such that

$$(4.16) \quad [\chi(x_{\pi^{-1}(0)}, \dots, x_{\pi^{-1}(n-1)})]_{\mathcal{M}} = [f(\llbracket \chi(\vec{x}) \rrbracket_{\mathcal{M}})]_{\mathcal{M}}$$

Alternatively,  $\pi$  must satisfy

$$(4.17) \quad \mathcal{M} \models \chi(x_{\pi^{-1}(0)}, \dots, x_{\pi^{-1}(n-1)}) \leftrightarrow f(\llbracket \chi(\vec{x}) \rrbracket_{\mathcal{M}})$$

We may write each formula as  $\varphi(x_0, \dots, x_{n-1})$  even if some of the variables do not appear in it. A formula may thus have several types, since nonoccurring variables can be permuted freely (also it may happen that a relation is symmetric in some columns). The type of a formula may not be unique, so define

$$(4.18) \quad \text{tp}(\varphi) := \{ \pi \in \Pi_n : \pi \text{ is a type of } \varphi \}$$

This is uniquely defined. Finally, for a given type  $\pi$  we define the linking aspect  $Y_\pi$  to be

$$(4.19) \quad Y_\pi(\llbracket \chi \rrbracket_{\mathcal{M}}) := \llbracket [x_{\pi(i)}/x_i : i < n] f(\llbracket \chi \rrbracket_{\mathcal{M}}) \rrbracket_{\mathcal{M}}$$

Together with (4.16) this gives us

$$(4.20) \quad [\chi(x_0, \dots, x_{n-1})]_{\mathcal{M}} = Y_\pi(\llbracket \chi \rrbracket_{\mathcal{M}})$$

This finishes the preparations. We are ready to spell out the modes. For the existential quantifier we introduce the following functions

$$(4.21) \quad \exists_\pi^i(c) := \llbracket \Pi_i \cdot Y_\pi(c) \rrbracket_{\mathcal{M}}$$

For the universal quantifier we use

$$(4.22) \quad \forall_{\pi}^i(c) := \langle\langle -\Pi_i \cdot - Y_{\pi}(c) \rangle\rangle_{\mathcal{M}}$$

where  $-$  is the relative complement. Now for the booleans. Let  $()$  denote the identity permutation.

$$(4.23) \quad \begin{aligned} N(c) &:= \langle\langle -Y_0(c) \rangle\rangle_{\mathcal{M}} \\ A_{\pi,\rho}(c, d) &:= \langle\langle Y_{\pi}(c) \vee Y_{\rho}(d) \rangle\rangle_{\mathcal{M}} \\ C_{\pi,\rho}(c, d) &:= \langle\langle Y_{\pi}(c) \wedge Y_{\rho}(d) \rangle\rangle_{\mathcal{M}} \\ I_{\pi,\rho}(c, d) &:= \langle\langle Y_{\pi}(c) \rightarrow Y_{\rho}(d) \rangle\rangle_{\mathcal{M}} \end{aligned}$$

The modes are as follows: for every relation symbol  $R$  and map  $i : \{0, 1, \dots, n-1\} \rightarrow \{0, 1, \dots, n-1\}$  (not necessarily injective) an 0-ary mode  $f_i^R$ . For every  $i < n$  and every  $\pi \in \Pi_n$  a unary mode  $f_{i,\pi}^{\exists}$  and a unary mode  $f_{i,\pi}^{\forall}$ . A unary mode  $f^{\neg}$  and for every  $\pi, \rho \in \Pi_n$  (not necessarily distinct) binary modes  $f_{\pi,\rho}^{\wedge}$ ,  $f_{\pi,\rho}^{\vee}$ , and  $f_{\pi,\rho}^{\rightarrow}$ . This defines the set  $F_n$  and the signature  $\Omega_n$ . Now we define the interpretation  $\mathcal{J}_n$ . (We use the subscript notation  $_i$  in place of the otherwise used notation using binary strings. Since the number of variables is finite, this is unproblematic.)

$$(4.24) \quad \begin{aligned} \mathcal{J}_n(f_i^R) &:= \langle R(\hat{\cdot} \mathbf{x}_{i(0)} \hat{\cdot}, \hat{\cdot} \cdots \hat{\cdot}, \hat{\cdot} \mathbf{x}_{i(a(R)-1)} \hat{\cdot}), \\ &\quad \langle\langle R(\hat{\cdot} \mathbf{x}_{i(0)} \hat{\cdot}, \hat{\cdot} \cdots \hat{\cdot}, \hat{\cdot} \mathbf{x}_{i(a(R)-1)} \hat{\cdot}) \rangle\rangle_{\mathcal{M}} \rangle \\ \mathcal{J}_n(f^{\neg})(\langle e, m \rangle) &:= \langle (\hat{\cdot} \neg \hat{\cdot} e \hat{\cdot}), N(m) \rangle \\ \mathcal{J}_n(f_{i,\pi}^{\exists})(\langle e, m \rangle) &:= \begin{cases} \langle (\hat{\cdot} (\exists \mathbf{x}_i \hat{\cdot}) \hat{\cdot} e, \exists_{\pi}^i(m)) \rangle & \text{if } \pi \in \text{tp}(e) \\ \text{undefined} & \text{else} \end{cases} \\ \mathcal{J}_n(f_{i,\pi}^{\forall})(\langle e, m \rangle) &:= \begin{cases} \langle (\hat{\cdot} (\forall \mathbf{x}_i \hat{\cdot}) \hat{\cdot} e, \forall_{\pi}^i(m)) \rangle & \text{if } \pi \in \text{tp}(e) \\ \text{undefined} & \text{else} \end{cases} \\ \mathcal{J}_n(f_{\pi,\rho}^{\vee})(\langle e, m \rangle, \langle e', m' \rangle) &:= \begin{cases} \langle (\hat{\cdot} e \hat{\cdot} \wedge \hat{\cdot} e' \hat{\cdot}), A_{\pi,\rho}(m, m') \rangle & \text{if } \pi \in \text{tp}(e) \\ & \text{and } \rho \in \text{tp}(e') \\ \text{undefined} & \text{else} \end{cases} \\ \mathcal{J}_n(f_{\pi,\rho}^{\wedge})(\langle e, m \rangle, \langle e', m' \rangle) &:= \begin{cases} \langle (\hat{\cdot} e \hat{\cdot} \vee \hat{\cdot} e' \hat{\cdot}), C_{\pi,\rho}(m, m') \rangle & \text{if } \pi \in \text{tp}(e) \\ & \text{and } \rho \in \text{tp}(e') \\ \text{undefined} & \text{else} \end{cases} \\ \mathcal{J}_n(f_{\pi,\rho}^{\rightarrow})(\langle e, m \rangle, \langle e', m' \rangle) &:= \begin{cases} \langle (\hat{\cdot} e \hat{\cdot} \rightarrow \hat{\cdot} e' \hat{\cdot}), I_{\pi,\rho}(m, m') \rangle & \text{if } \pi \in \text{tp}(e) \\ & \text{and } \rho \in \text{tp}(e') \\ \text{undefined} & \text{else} \end{cases} \end{aligned}$$

**Theorem 4.1** *The grammar  $G_n = \langle \Omega_n, \mathcal{J}_n \rangle$  is independent and  $L(G_n) = L_n$ .*

**Proof.** It is easy to see that  $G_n$  is independent. The functions on the concepts are defined, and the functions on the exponents are partial, with conditions that are completely independent of the meaning. (This is because the concept of a formula is uniquely determined anyway, so any mention of meaning of a sign can be eliminated.) It remains to be shown that the grammar generates  $L_n$ . First, for every  $\varphi$  the grammar does generate a sign of the form  $\langle \varphi, m \rangle$  for some  $m$ . This is shown by induction. The base case is

$$(4.25) \quad \varphi = R(x_{j_0}, \dots, x_{j_{a(n)-1}})$$

Put  $j(k) := j_k$  if  $k < a(R)$  and  $j(k) := 0$  else. Then

$$(4.26) \quad \varphi = R(x_{i(0)}, \dots, x_{i_{(a(n)-1)}})$$

and is generated by  $f_i^R$ . As for the inductive steps, suppose for example that the formula has the form  $(e \wedge e')$ . By inductive hypothesis there are analysis terms  $t$  and  $t'$  that yield (at least) the exponents  $e$  and  $e'$ . Let  $\pi$  be a type of  $e$  and  $\rho$  a type of  $e'$ . (Every formula has at least one type.) Then  $f_{\pi, \rho}^{\wedge} t t'$  is defined and has exponent  $(e \wedge e')$ . The other cases are similar.

It remains to be shown that for every  $\varphi$ , we get only the sign  $\langle \varphi, \llbracket \varphi \rrbracket \mathcal{M} \rangle$ . Again this is done by induction. The base cases are valid by definition. We treat two inductive cases only, but they are general enough. The first is  $f_{i, \pi}^{\exists}$ . So, suppose we have generated the sign  $\langle \chi, m \rangle$  using the term  $t$ . The induction hypothesis is that  $m = \llbracket \chi \rrbracket \mathcal{M}$ . Assume that  $\chi$  has type  $\pi$ . Then  $f_{i, \pi}^{\exists}$  can be applied to the sign and we get

$$(4.27) \quad \mathcal{J}_n(f_{i, \pi}^{\exists})(\langle \chi, m \rangle) = \langle (\wedge \exists x_i \wedge e \wedge), \exists_{\pi}^i(m) \rangle$$

What remains to be shown is that

$$(4.28) \quad \exists_{\pi}^i(m) = \llbracket (\exists x_i) \chi \rrbracket \mathcal{M}$$

By (4.21) and the hypothesis that  $m = \llbracket \chi \rrbracket \mathcal{M}$  it is enough to show that

$$(4.29) \quad \llbracket (\exists x_i) \chi \rrbracket \mathcal{M} = \llbracket \Pi_i . Y_{\pi}(\llbracket \chi \rrbracket) \rrbracket$$

on the basis that  $\chi$  has type  $\pi$ . To do that it is certainly enough to establish

$$(4.30) \quad \llbracket (\exists x_i) \chi \rrbracket \mathcal{M} = \Pi_i . Y_{\pi}(\llbracket \chi \rrbracket)$$

This however immediately follows from (4.20). Now assume that the signs  $\langle e, m \rangle$  and  $\langle e', m' \rangle$  are generated, using analysis terms  $t$  and  $t'$ . Let  $e$  have type  $\pi$  and  $e'$  type  $\rho$ . By inductive hypothesis,  $m = \langle e \rangle_{\mathcal{M}}$  and  $m' = \langle e' \rangle_{\mathcal{M}}$ . Then  $f_{\pi, \rho}^{\vee} t t'$  is defined and has exponents  $(e \vee e')$ . Then for the meanings we have

$$\begin{aligned}
 & A_{\pi, \rho}(m, m') \\
 (4.31) \quad & = \llbracket Y_{\pi}(m) \cup Y_{\rho}(m') \rrbracket \\
 & = \llbracket [e]_{\mathcal{M}} \cup [e']_{\mathcal{M}} \rrbracket \\
 & = \langle \langle e \vee e' \rangle \rangle_{\mathcal{M}}
 \end{aligned}$$

Similarly for the other modes. □

Now we wish to turn to the question which operations may be useful for natural languages. The easiest operation is the product. Let  $R \in \mathfrak{c}$  and  $S \in \mathfrak{d}$ . Then put

$$(4.32) \quad \mathfrak{c} \times \mathfrak{d} := \llbracket R \times S \rrbracket$$

This is independent of the choice of representing relations. Now let  $Y$  be a linking aspect. Based on this aspect we may define the operation

$$(4.33) \quad Y^{[j]}(\mathfrak{c}) := \llbracket \Pi_j . Y(\mathfrak{c}) \rrbracket$$

Here, if  $R$  is of length  $n$ ,

$$(4.34) \quad \Pi_j . R = \{ \langle a_0, \dots, a_{j-1}, a_{j+1}, \dots, a_{n-1} \rangle : \text{exists } a_j : \langle a_0, \dots, a_{n-1} \rangle \in R \}$$

This is a relation of length  $n - 1$ . This is the existential quantification of the variable  $j$  in the relation  $Y(\mathfrak{c})$ . If  $Y(\mathfrak{c})$  has length  $< j$  this is  $\mathfrak{c}$  again. Another operation is this. Let  $d_{jk}^n$  be the set of tuples of length  $n$  such that  $a_j = a_k$ . Then

$$(4.35) \quad Y_{[j=k]}(\mathfrak{c}) := \llbracket Y(\mathfrak{c}) \cap d_{jk}^n \rrbracket$$

It will be useful to combine the previous operations in one step:

$$(4.36) \quad Y_{[j=k; j'=k'; \dots; j^q=k^q]}^{[i; i'; \dots; i^p]}(\mathfrak{c}) := \llbracket \Pi_i . \Pi_{i'} . \dots . \Pi_{i^p} . (Y(\mathfrak{c}) \cap d_{jk}^n \cap d_{j'k'}^n \cap \dots \cap d_{j^q k^q}^n) \rrbracket$$

### 4.3 A Fragment of English

In this section we shall show by way of examples in which way one can overcome the limitations of the semantics. The first strategy is to use thematic roles. The

idea is that in an event of some sort the participants can be distinguished by some property that they have as opposed to the others. For example, the meaning of the verb *hit* may—in standard notation—be a relation  $\text{hit}'(t, w, x, y)$  where  $t$  is a time point,  $w$  is a possible world or situation, and  $x$  and  $y$  are things. In this case it is already possible to distinguish the variable  $t$  from the others due to the fact that all variables are sortal. A times variable can never be identical to a world variable or an entity variable; and the things that these variables denote are completely separate, too. Likewise  $w$  is singled out through the sort. However,  $x$  and  $y$  are sortally identical. Nevertheless, we can distinguish them by observing that in an act of hitting there is one participant that exerts force on the other. It is this one that performs an action, while the other can be completely at rest. Thus, there is a formula  $\alpha(t, w, x)$  such that

$$(4.37) \quad \vdash \text{hit}'(t, w, x, y) \rightarrow \alpha(t, w, x), \quad \not\vdash \text{hit}'(t, w, x, y) \rightarrow \alpha(t, w, y)$$

This is essentially the theory proposed by [Wechsler, 1995]. Wechsler uses modal notation, so it would look more like

$$(4.38) \quad \vdash \Box(\text{hit}'(x, y) \rightarrow \alpha(x)), \quad \not\vdash \Box(\text{hit}'(x, y) \rightarrow \alpha(y))$$

But these differences are superficial. Let us suppose that something like (4.37) holds. Since we are not talking about logical truth, we are rather talking about a fixed interpretation, we should rather require the following (with  $\pi_{(23)}$  the permutation interchanging the third and the fourth column; for better readability I write  $\pi[\varphi]_{\mathcal{M}}$  in place of the more correct  $\pi[[\varphi]_{\mathcal{M}}]$ ):

$$(4.39) \quad \begin{aligned} & [\text{hit}'(t, w, x, y)]_{\mathcal{M}} \subseteq [\alpha(t, w, x)]_{\mathcal{M}} \times M_e \\ & \pi_{(23)}[\text{hit}'(t, w, x, y)]_{\mathcal{M}} \not\subseteq [\alpha(t, w, x)]_{\mathcal{M}} \times M_e \end{aligned}$$

The formula  $\alpha(t, w, x)$  does not suffer from the same combinatorial ambiguity. Thus, the concept  $\langle\langle \alpha(t, w, x) \rangle\rangle_{\mathcal{M}}$  has only *one* minimal member. The task of picking out the correct representative has become trivial. So, we pick the minimal member  $R$  and then return to  $\text{hit}'(t, w, x, y)$ . The concept has two minimal members, say  $S$  and  $T$ . According to the above, we have  $R \times M_e \subseteq S$  and  $R \times M_e \not\subseteq T$  or  $R \times M_e \not\subseteq S$  and  $R \times M_e \not\subseteq T$ . Thus, there is a way to find out which minimal member to pick.

**Example 14.** There are three sorts,  $e$ ,  $w$  and  $t$ . Assume that  $M_e = \{a, b, c\}$ ,  $M_w = \{w_0, w_1\}$ , and  $M_t = \{t_0, t_1\}$ .

$$(4.40) \quad [\alpha(t, w, x)]_{\mathcal{M}} = \{\langle t_0, w_0, a \rangle, \langle t_0, w_0, b \rangle, \langle t_0, w_0, c \rangle, \langle t_0, w_1, a \rangle, \\ \langle t_1, w_0, b \rangle, \langle t_1, w_0, c \rangle\}$$

$$(4.41) \quad [\text{hit}'(t, w, x, y)]_{\mathcal{M}} = \{\langle t_0, w_0, a, a \rangle, \langle t_0, w_0, a, b \rangle, \langle t_0, w_0, b, a \rangle, \\ \langle t_0, w_0, a, c \rangle, \langle t_1, w_0, c, a \rangle, \langle t_1, w_0, c, b \rangle\}$$

In this model (4.39) is satisfied. This means that we can discriminate the two minimal members of the concept:

$$(4.42) \quad \llbracket \text{hit}'(t, w, x, y) \rrbracket_{\mathcal{M}} = \{\{\langle t_0, w_0, a, a \rangle, \langle t_0, w_0, a, b \rangle, \langle t_0, w_0, b, a \rangle, \\ \langle t_0, w_0, a, c \rangle, \langle t_1, w_0, c, a \rangle, \langle t_1, w_0, c, b \rangle\}, \\ \{\langle t_0, w_0, a, a \rangle, \langle t_0, w_0, b, a \rangle, \langle t_0, w_0, a, b \rangle, \\ \langle t_0, w_0, c, a \rangle, \langle t_1, w_0, a, b \rangle, \langle t_1, w_0, b, c \rangle\}, \dots\}$$

Indeed, the second member contains  $\langle t_1, w_0, a, b \rangle$ , and this is not contained in the set  $[\alpha(t, w, x)]_{\mathcal{M}} \times M_e$ .

$$(4.43) \quad [\alpha(t, w, x)]_{\mathcal{M}} \times M_e = \{\langle t_0, w_0, a, a \rangle, \langle t_0, w_0, a, b \rangle, \langle t_0, w_0, a, c \rangle, \\ \langle t_0, w_0, b, a \rangle, \langle t_0, w_0, b, b \rangle, \langle t_0, w_0, b, c \rangle, \langle t_0, w_0, c, a \rangle, \\ \langle t_0, w_0, c, b \rangle, \langle t_0, w_0, c, c \rangle, \langle t_0, w_1, a, a \rangle, \langle t_0, w_1, a, b \rangle, \\ \langle t_0, w_1, a, c \rangle, \langle t_1, w_0, b, a \rangle, \langle t_1, w_0, b, b \rangle, \langle t_1, w_0, b, c \rangle, \\ \langle t_1, w_0, c, a \rangle, \langle t_1, w_0, c, b \rangle, \langle t_1, w_0, c, c \rangle\}$$

Notice that in  $w_0$  at  $t_0$  every object has property  $\alpha$ , so there would be no way to distinguish the arguments. For example, suppose that in this world and at this time everybody is such that he or she is moving and exerting some force. Still it does not follow that everybody is hitting someone. They could, for example, push a car uphill. However, if (4.39) is to make sense at all then there should at least be some world and some time point where we find that some entity  $u$  hits some entity  $v$ , and  $v$  does not possess  $\alpha$ . ♣

We shall display a primitive grammar. It has five modes:  $F = \{f_0, f_1, f_2, f_3, f_4\}$ .  $\Omega(f_0) = \Omega(f_1) = \Omega(f_2) := 0$ ,  $\Omega(f_3) := \Omega(f_4) := 2$ . For the purpose of the next definition, let  $\sigma = \langle e, c, m \rangle$  and  $\sigma' = \langle e', c', m' \rangle$ . Further, let  $d_i^k j$  be the relation  $\{\langle a_0, \dots, a_{k-1} \rangle : a_i = a_j\}$ . (This relation is only defined if sorts match. For simplicity we suppress mentioning sorts.)  $Y$  is a linking aspect based that extends

the aspect in the previous example. What is important below is only that it orders the arguments like this: time, world, patient, actor.

(4.44)

$$\begin{aligned}
\mathcal{D}(f_0)() &:= \langle \text{John}, \text{NP}, \{a\} \rangle \\
\mathcal{D}(f_1)() &:= \langle \text{Paul}, \text{NP}, \{b\} \rangle \\
\mathcal{D}(f_2)() &:= \langle \text{hits}, \text{V}, \langle \text{hit}'(t, w, x, y) \rangle_{\mathcal{M}} \rangle \\
\mathcal{D}(f_3)(\sigma, \sigma') &:= \begin{cases} \langle e' \sqcap e', \text{VP}, \langle \Pi_2. \Pi_4. Y(m) \times Y(m') \cap d_{24}^5 \rangle_{\mathcal{M}} \rangle \\ \quad \text{if } c = \text{V and } c' = \text{NP} \\ \text{undefined else} \end{cases} \\
\mathcal{D}(f_4)(\sigma, \sigma') &:= \begin{cases} \langle e' \sqcap e' ., \text{S}, \langle \Pi_0. \Pi_1. \Pi_2. \Pi_3. (Y(m) \times Y(m') \cap d_{23}^4) \rangle_{\mathcal{M}} \rangle \\ \quad \text{if } c = \text{VP and } c' = \text{NP} \\ \text{undefined else} \end{cases}
\end{aligned}$$

The resulting meaning of a sentence is true if there is a time point and world such that the sentence is true in that world at that time. (We shall improve that soon.) Let us see how that goes. Then sentence **John hits Paul.** can be generated only as the exponent of  $f_4 f_3 f_2 f_1 f_0$ . Let us do this step by step.

$$\begin{aligned}
\iota_G(f_3 f_2 f_1) &= \mathcal{D}(f_3)(\langle \text{hits}, \text{V}, R \rangle, \langle \text{Paul}, \text{NP}, \{b\} \rangle) \\
(4.45) \quad &= \langle \text{hits} \sqcap \text{Paul}, \text{VP}, \langle \Pi_3. Y(m) \times Y(m') \cap d_{34}^4 \rangle_{\mathcal{M}} \rangle \\
&= \langle \text{hits Paul}, \text{VP}, \langle \{ \langle t_0, w_0, a \rangle, \langle t_1, w_0, c \rangle \} \rangle_{\mathcal{M}} \rangle
\end{aligned}$$

Here is how the concept in the last step is derived. First, we apply the linking aspect  $Y$  to the concept of hitting, whereupon we get

$$\begin{aligned}
(4.46) \quad Y(m) &= \{ \langle t_0, w_0, a, a \rangle, \langle t_0, w_0, b, a \rangle, \langle t_0, w_0, a, b \rangle, \\
&\quad \langle t_0, w_0, c, a \rangle, \langle t_1, w_0, a, b \rangle, \langle t_1, w_0, b, c \rangle \}
\end{aligned}$$

Also,  $Y(m') = \{b\}$ , since there is nothing to order. We take the product:

$$\begin{aligned}
(4.47) \quad Y(m) \times Y(m') &= \{ \langle t_0, w_0, a, a, b \rangle, \langle t_0, w_0, b, a, b \rangle, \langle t_0, w_0, a, b, b \rangle, \\
&\quad \langle t_0, w_0, c, a, b \rangle, \langle t_1, w_0, a, b, b \rangle, \langle t_1, w_0, b, c, b \rangle \}
\end{aligned}$$

Next we intersect with the set  $d_{24}^5$ . That is to say we take the subset of all vectors  $\langle x_0, x_1, x_2, x_3, x_4 \rangle$  such that  $x_2 = x_4$ .

$$(4.48) \quad Y(m) \times Y(m') \cap d_{24}^5 = \{ \langle t_0, w_0, b, a, b \rangle, \langle t_1, w_0, b, c, b \rangle \}$$

Finally, we remove the columns 2 and 4:

$$(4.49) \quad \Pi_2.\Pi_4.Y(m) \times Y(m') \cap d^{24} = \{\langle t_0, w_0, a \rangle, \langle t_1, w_0, c \rangle\}$$

And then we form the concept, which just means that we forget the order of the columns. Call that concept  $m$ . We are ready to continue:

$$(4.50) \quad \begin{aligned} \iota_G(f_4.f_3.f_2.f_1.f_0) &= \mathcal{D}(f_4)(\iota_G(f_3.f_2.f_1), \iota_G(f_0)) \\ &= \mathcal{D}(f_4)(\langle \text{hits Paul, VP}, \ll \langle t_0, w_0, a \rangle, \langle t_1, w_0, c \rangle \gg_{\mathcal{M}} \rangle, \\ &\quad \langle \text{John, NP}, \{a\} \rangle) \\ &= \langle \text{John} \hat{\square} \text{hits Paul} \hat{\cdot}, \text{S}, \\ &\quad \ll \Pi_0.\Pi_1.\Pi_2.\Pi_3.(Y(m) \times Y(\ll \{a\} \gg_{\mathcal{M}}) \cap d_{23}^4) \gg_{\mathcal{M}} \rangle \\ &= \langle \text{John hits Paul} \hat{\cdot}, \text{S}, \{\emptyset\} \rangle \end{aligned}$$

The way to get there is as follows. The linking aspect orders the minimal members of the concept  $m$ . Assume that it does that on the basis times  $\downarrow$  worlds  $\downarrow$  entities. (This does not follow, by the way, from our assumption on how it orders the minimal members of the concept of hitting!) Then

$$(4.51) \quad Y(m) = \{\langle t_0, w_0, a \rangle, \langle t_1, w_0, c \rangle\}$$

It also orders the unique minimal member of the concept of John and gives us  $\{a\}$ . We take the product

$$(4.52) \quad Y(m) \times Y(\ll \{a\} \gg_{\mathcal{M}}) = \{\langle t_0, w_0, a, a \rangle, \langle t_1, w_0, c, a \rangle\}$$

Next we intersect with  $d_{23}^4$ :

$$(4.53) \quad Y(m) \times Y(\ll \{a\} \gg_{\mathcal{M}}) \cap d_{23}^4 = \{\langle t_0, w_0, a, a \rangle\}$$

And then we eliminate the columns 0, 1, 2, and 3:

$$(4.54) \quad \Pi_0.\Pi_1.\Pi_2.\Pi_3.Y(m) \times Y(\ll \{a\} \gg_{\mathcal{M}}) \cap d_{23}^4 = \{\langle \rangle\}$$

We use the convention  $\langle \rangle := \emptyset$ . The sentence is true in the model.

When we move to more complex cases, for example relations involving 3 entities (arising in the meaning of ditransitives, for example) we do not need to come up with an  $\alpha$  such that, say

$$(4.55) \quad \begin{aligned} [\varphi(t, w, x, y, z)]_{\mathcal{M}} &\subseteq [\alpha(t, w, x)]_{\mathcal{M}} \times M_e \times M_e \\ \pi_{(23)}[\varphi(t, w, x, y, z)]_{\mathcal{M}} &\not\subseteq [\alpha(t, w, y)]_{\mathcal{M}} \times M_e \times M_e \\ \pi_{(24)}[\varphi(t, w, x, y, z)]_{\mathcal{M}} &\not\subseteq [\alpha(t, w, z)]_{\mathcal{M}} \times M_e \times M_e \end{aligned}$$

It is enough if we first find a concept that allows to separate two variables from a third and then continue as before.

The formulae above do not always exist. A case in point is the relation  $<$ . If taken as a relation on the natural numbers, we can use the formula  $\alpha(y) := (y \neq 0)$ . For there is no  $x$  such that  $x < 0$ , it is through this property that we can discriminate the positions. However, matters change when we look at it as a relation between integers. For the projection of  $<$  onto both of its components is the set  $\mathbb{Z}$  of integers. This means that for every  $x$  there is a number  $y$  that is bigger than  $x$ , and for every  $y$  there is a number  $x$  that is smaller than  $y$ . Thus we have to use a different tool. One idea that actually always works is this.

**Definition 4.2** A *sampler* is a function  $\mathbb{S}$  from concepts to finite sets of tuples such that if  $c$  is a concept, then there is exactly one minimal  $R \in c$  with  $R \supseteq \mathbb{S}(c)$ .

Samplers always exist. For let  $c$  be a concept; fix a minimal member  $c$  of  $R$ . Let  $\Xi$  be the set of permutations such that  $\pi[R] \neq R$ . (In fact, we can skip all permutations that are not sortally correct. Here, a permutation  $\pi$  is sortally correct if for the sequence  $\vec{s}$  or sorts:  $\pi(\vec{s}) = \vec{s}$ .) For every  $\pi \in \Xi$  pick a tuple  $\vec{x}_\pi$  such that  $\vec{x}_\pi \in R$  but  $\vec{x}_\pi \notin \pi[R]$ . By assumption for every  $\pi \in \Xi$  such a tuple exists. Let

$$(4.56) \quad \mathbb{S}(c) := \{\vec{x}_\pi : \pi \in \Xi\}$$

If we want to use a sampler to pick out a different minimal member  $U$  from  $c$ , then since that member is a permutation of the original set  $R$ , say  $U = \rho[R]$ , we can use in place of  $\mathbb{S}(c)$  the set  $\rho(\mathbb{S}(c))$ .

**Example 15.** In the example above, the following is a sampler for  $\langle \text{hit}'(t, w, x, y) \rangle_{\mathcal{M}}$  picking out  $R := [\text{hit}'(t, w, x, y)]_{\mathcal{M}}$ : it is  $\{\langle t_0, w_0, a, c \rangle\}$ . This is because the only permutations that are sortally correct are the identity  $\pi_0$  and  $\pi_{(23)}$ . Thus,  $\Xi := \{\pi_{(23)}\}$  is enough. For the permutation  $\pi_{(23)}$  we have  $\pi_{(23)}(\langle t_0, w_0, a, c \rangle) = \langle t_0, w_0, c, a \rangle$  which is not in the relation. The set  $\{\langle c, a, t_0, w_0 \rangle\}$  instead picks out the member  $\pi_{(0213)}[R]$ , or if you will, the set  $[\text{see}'(y, x, t, w)]_{\mathcal{M}}$ . ♣

**Example 16.** Assume one sort  $e$ , and  $M_e = \{a, b, c\}$ . Let

$$(4.57) \quad R = \{\langle a, b, c \rangle, \langle a, c, b \rangle, \langle b, a, b \rangle, \langle b, b, a \rangle\}$$

Then it turns out that  $\Xi = \{\pi_{(01)}, \pi_{(02)}\}$ , because the permutation (12) transforms  $R$  into itself. To fix  $\langle R \rangle_{\mathcal{M}}$  to  $R$ , we use  $\{\langle a, b, c \rangle\}$ . ♣

## 4.4 Why Use Concepts?

It seems that the introduction of concepts actually made matters worse. To get compositional definitions is not a straightforward matter. When we compare that with other approaches (Montague Grammar, or DRT based approaches such as [Kamp and Reyle, 1993]) we ask ourselves whether it is really warranted to replace, say, DRSs by concepts. To see that one is virtually compelled to assume concepts, look at the way the algorithms of [Kamp and Reyle, 1993] factually do. They translate the sentence (4.58) not directly, but via surface indexing.

(4.58) A big man sees a small cat.

A surface indexing is an assignment of indices to the free variables of the corresponding DRS. Such indices were once assumed to be distributed by the parser in terms of annotations to the words of the surface string. Thus the input to the translation algorithm is (4.59) rather than (4.58).<sup>1</sup>

(4.59)  $A_1$  big<sub>1</sub> man<sub>1</sub> sees<sub>(1,7)</sub> a<sub>7</sub> small<sub>7</sub> cat<sub>7</sub>.

Based on the input the translation is unique. The problem with this notion of syntax is that it uses material that is not in the actual surface string, namely indices. The indices in turn determine the translation into a DRS, or for that matter, into some predicate logical formula. It turns out that  $\text{man}_0$  has a different translation from  $\text{man}_1$ . Therefore, in order for the proposed algorithm to work, we must assume that the grammar generates entries of the following form:

(4.60)  $\langle \text{man}_0, \text{man}'(x_0) \rangle, \langle \text{man}_1, \text{man}'(x_1) \rangle, \langle \text{man}_2, \text{man}'(x_2) \rangle, \dots$

It does not necessarily mean that the above entries are in the lexicon. For the indices may be taken to be, say, decimal strings; in that case we need a base entry

(4.61)  $\langle \text{man}_0, \text{man}'(x) \rangle$

---

<sup>1</sup>Note that the indices are also written using typewriter fonts. This highlights the fact that they are really there, and they also have to be written using some characters of the alphabet.

and ten unary functions (to append a digit to the index) to successfully generate all of these entries.

For a transitive verb we will have

$$\begin{aligned}
 (4.62) \quad & \langle \text{sees}_{(0,0)}, \text{see}'(x_0, x_0) \rangle, \langle \text{sees}_{(1,0)}, \text{see}'(x_1, x_0) \rangle, \\
 & \langle \text{sees}_{(2,0)}, \text{see}'(x_2, x_0) \rangle, \dots \\
 & \langle \text{sees}_{(0,1)}, \text{see}'(x_0, x_1) \rangle, \langle \text{sees}_{(1,1)}, \text{see}'(x_1, x_1) \rangle, \\
 & \langle \text{sees}_{(2,1)}, \text{see}'(x_2, x_1) \rangle, \dots \\
 & \langle \text{sees}_{(0,2)}, \text{see}'(x_0, x_2) \rangle, \langle \text{sees}_{(1,2)}, \text{see}'(x_1, x_2) \rangle, \\
 & \langle \text{sees}_{(2,2)}, \text{see}'(x_2, x_2) \rangle, \dots \\
 & \dots \qquad \qquad \qquad \dots
 \end{aligned}$$

This is where our principles come in. Recall that we have explicitly ruled out deletion. If there is no index on the surface, there has never been one in the beginning. So, on the deep phonological level we also have just *man* and *sees*.<sup>2</sup> Thus we rather have the following signs

$$\begin{aligned}
 (4.63) \quad & \langle \text{man}, \text{man}'(x_0) \rangle, \langle \text{man}, \text{man}'(x_1) \rangle, \langle \text{man}, \text{man}'(x_2) \rangle, \dots \\
 (4.64) \quad & \langle \text{sees}, \text{see}'(x_0, x_0) \rangle, \langle \text{sees}, \text{see}'(x_1, x_0) \rangle, \langle \text{sees}, \text{see}'(x_2, x_0) \rangle, \dots \\
 & \langle \text{sees}, \text{see}'(x_0, x_1) \rangle, \langle \text{sees}, \text{see}'(x_1, x_1) \rangle, \langle \text{sees}, \text{see}'(x_2, x_1) \rangle, \dots \\
 & \langle \text{sees}, \text{see}'(x_0, x_2) \rangle, \langle \text{sees}, \text{see}'(x_1, x_2) \rangle, \langle \text{sees}, \text{see}'(x_2, x_2) \rangle, \dots \\
 & \dots \qquad \qquad \qquad \dots \qquad \qquad \qquad \dots
 \end{aligned}$$

This means that the name of the actual variable has become immaterial. This is essentially what is meant by *alphabetical innocence* (see [Fine, 2003]).

Let us spell out what this effectively means for us. Let us assume for simplicity that we are within a finite variable fragment; then the meaning of a predicate logical formula (with just one type) is a subset of  $M^n$ ,  $M$  the underlying set. Here, the sequence  $\langle a_0, a_1, \dots, a_{n-1} \rangle$  is saying that the variable  $x_0$  is assigned the object  $a_0$ ,  $x_1$  the object  $a_1$ , and so on. If we now assume that we may freely change the indexing, we get as a result the following:

<sup>2</sup>Given that we allow compositionality at the deep phonological level and not the surface it might be deemed that we only need to propose a regular relation that deletes the indices. However, such an operation lacks any phonological motivation. In particular, since the symbols we use (smaller font size lowered numbers) do not appear in ordinary language, their use is ruled out by the fact that none of the symbols actually exists in the language itself. It is therefore excluded.

**Principle 6 (Alphabetical Innocence)** *Suppose a formula  $\varphi$  represents the meaning of a natural language string. Let  $s$  be a substitution that is injective on the variables of  $\varphi$ ; and let  $s(\varphi)$  be the result of replacing every occurrence of  $x_i$  by  $s(x_i)$ . Then  $s(\varphi)$  is equivalent to  $\varphi$ .*

This principle however is highly specific; it talks about natural language strings without specifying what that actually is. There is no notion of natural language that we have isolated that will allow us to say which languages are natural and which ones are not. Thus, the principle of alphabetical innocence will have to follow from something else that does not use that term.

**Principle 7 (Invariance)** *There are no two lexical entries of the form  $\langle e, m \rangle$  and  $\langle e', m' \rangle$  with  $e = e'$  and  $m \neq m'$ , except if  $m$  and  $m'$  are of different sort.*

This says that if  $m$  and  $m'$  can be joined into one meaning (and they can if they are of the same sort) then they should be. This presupposes that the meanings can be unified (that is, meanings must have a set like structure), but this is typically given. Alternatively, we may comply with it by saying that all the meanings that we proposed for a given exponent must be equivalent. This principle prohibits the proliferation of lexical signs. Applied to (4.63) and (4.64) this may either mean that *man* *simultaneously* means  $\text{man}'(x_0, \text{man}'(x_1)$  and so on. So, it may mean that we have the following sign (given one has infinite disjunctions):

$$(4.65) \quad \langle \text{man}, \bigvee_{i \in \mathbb{N}} \text{man}'(x_i) \rangle$$

However, we preferred to make all these meanings equivalent. Thus we are led to assume *Alphabetic Innocence*. The meaning equivalence makes us to assume that for a permutation  $\pi \in \Pi_n$ :

$$(4.66) \quad \varphi(x_0, x_1, \dots, x_{n-1}) \sim \varphi(x_{\pi(0)}, x_{\pi(1)}, \dots, x_{\pi(n-1)})$$

We may go one step further and say that we also have the following signs:

$$(4.67) \quad \langle \text{man}, \text{man}'(x_0) \wedge x_0 = x_3 \rangle, \quad \langle \text{man}, \text{man}'(x_1) \wedge x_0 = x_0 \rangle$$



# Bibliography

- [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.
- [Falk, 2001] Yehuda Falk. *Lexical-Functional Grammar: An Introduction to Parallel Constraint-Based Syntax*. CSLI, Stanford, 2001.
- [Fine, 2003] Kit Fine. The Role of Variables. *Journal of Philosophy*, 50:605 – 631, 2003.
- [Kamp and Reyle, 1993] Hans Kamp and Uwe Reyle. *From Discourse to Logic. Introduction to Modeltheoretic Semantics of Natural Language, Formal Language and Discourse Representation*. Kluwer, Dordrecht, 1993.
- [Kracht, 2003] Marcus Kracht. *The Mathematics of Language*. Mouton de Gruyter, Berlin, 2003.
- [Lamb, 1966] Sydney M. Lamb. *Outline of Stratificational Grammar*. Georgetown University Press, Washington, 1966.
- [Mel'čuk, 2000] Igor Mel'čuk. *Cours de Morphologie Générale. (General Morphology. A Coursebook)*, volume 1 – 5. Les Presses de l'Université de Montréal, 2000.
- [Pentus, 1997] Mati Pentus. Product-Free Lambek-Calculus and Context-Free Grammars. *Journal of Symbolic Logic*, 62:648 – 660, 1997.
- [Pollard and Sag, 1994] Carl Pollard and Ivan Sag. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago, 1994.

- [Saussure, 1965] Ferdinand de Saussure. *Course in General Linguistics*. McGraw–Hill, Columbus, Ohio, 1965.
- [Wechsler, 1995] Stephen Wechsler. *The Semantic Basis of Argument Structure*. Dissertations in Linguistics. CSLI Publications, Stanford, 1995.