

# Lectures on Interpreted Languages and Compositionality

Marcus Kracht  
*Fakultät LiLi*  
*Universität Bielefeld*  
*Postfach 10 01 31*  
*33501 Bielefeld*  
marcus.kracht@uni-bielefeld.de

June 1, 2010



## Introduction

This manuscript presents an outline of something which I like to call *metatheory of linguistics*. It is not the attempt to replace any existing framework by a new one; it is rather the attempt to provide some result that show us the interconnection between certain requirements on theories. The word “metatheory” emphasises that we do not try to establish a new framework or discover concrete properties of languages but that we want to find methods of establishing the properties that a given language has. The aim is to find out in what way our initial assumptions 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 principles and investigate their empirical potential in this way. One such principle is the *Principle of Compositionality*. 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 that there can be concrete results about the structure of natural languages.

This book owes much to [Keenan and Stabler, 2001]. However, here the emphasis is quite different. Rather than assuming a particular grammar for a language at the outset, it is our aim to establish to what extent a language determines the grammar that generates it. In contrast to a lot of work in linguistics we do not take syntax as the exclusive source of evidence for structure, but both syntax and semantics *together*. Certainly, structural properties determine in which way expressions can be formed, but it has often been assumed in the linguistic literature that this is effectively all there is to be said about structure. This prejudice is the rather unfortunate heritage of a view promoted mainly—but not exclusively—within generative grammar. That linguistics is neither just about form (syntax) nor just about content (semantics) has been emphasised also by [Manaster-Ramer and Michalove, 2001] in the context of historical linguistics. A reconstruction solely based on either sound or meaning is useless. It must obviously involve both.

Although the style of the book is ostensibly neutral, the motivation for this research is the belief that ultimately the Principle of Compositionality is correct. However the methodology is not to try and verify it (this is impossible) but to see what consequences there are to the belief that it is true. For it turns out to be

possible to show that there are noncompositional languages. This means that it is an empirical question whether natural languages are compositional. It should be clear though that no definitive answer can be given for any natural language. The reason for this is a—in my view unavoidable—peculiarity of the whole theory; namely that for a finite language no grammar is needed. A simple enumeration of all items is enough. Hence, we can only fruitfully apply the theory to infinite languages. Thus, when we apply the present theory to a particular language we have to make assumptions about its global nature; and these assumptions are always tentative.

One of the biggest problem areas that I have identified in the course of studying compositionality is the nature of semantics. While the prevailing attitude used to be that meanings are hopelessly unclear, many semanticists nowadays feel that there is not much to discuss either: meanings are objects of the typed universe. Both groups will feel that the present book got it wrong; the first because I include semantics as primitive data, the second because I reject most semantic approaches to compositionality on the grounds that their semantics encodes semantically contingent syntactic structure. My response to the first group is this: if it is possible that humans understand each other, and if we do agree that there is such a thing as meaning, which can be preserved—among other—in translation, we must conclude that something of substance can be said about meanings, both concrete and abstract. The response to the second group is more complex. On the one hand, Montague Grammar has enjoyed a success, and it popularised the notion of compositionality. Nevertheless, I feel that there is a sort of complacency in most research conducted within type logical grammar as a whole. Most questions of actual meaning are not really solved, they are only relegated (for example to lexicology). Instead, much formal semantics is just offering technology without much proof that this is what we really wanted. It is much like saying that technological advances have made it possible for man to fly. That is only half true because originally the dream was to fly like a bird. It will take large parts of this book (especially Chapter 4) to work out exactly what is at fault with type theoretical semantics for natural language.

The Principle of Compositionality can be seen as an abstract requirement on the grammar of a language (and therefore, albeit indirectly, on the language itself). The rationale for adopting it, however, comes from an assumption on the architecture of language that is not universally shared. A particularly blatant case of this sort is generative grammar, where interpretation is done *after* the structure

building has taken place. It will be seen, though, that even if we grant this assumption, there is still so much to take care of that it becomes unclear just why a level such as LF is at all needed and how it can help us. In addition, it turns out that many more frameworks or theories are not compositional. This may be surprising since linguists commonly 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 what this book is 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 it rests. The actual syntactic structure, for example, has become highly theory internal in generative grammar. The independent evidence of Kayne's Antisymmetry Thesis, for example, was originally quite thin. And it is simply not true that it has been proved to be correct thereafter. Rather, the factual approach has been to adopt it and explore its consequences (just as I adopt here the Principle of Compositionality and explore its consequences). It is therefore not surprising that a careful review of the syntactic structure of German (within the generative framework) undertaken in [Sternefeld, 2006] has yielded a far less articulated structure than standardly assumed.

This book has basically two parts. The first consists in the Chapters 2 and 3, the second in Chapters 4 and 5. The first part develops a mathematical theory of interpreted languages; Chapter 2 provides the background of string languages, using grammars that generate the languages from the lexicon, known from Montague Grammar. Chapter 3 then turns to interpreted languages. In the second part, starting with Chapter 4 we zoom in on natural languages. We ask what the meanings of natural language constituents are and how they can be manipulated. Then, in Chapter 5 we apply the theory. We shall show that the notion of a concept defined in Chapter 4 changes the outlook on predicate logic: finite variable fragments are compositional, while with infinite variables the languages have no compositional context free grammar. Then we show how we can argue for structure from a purely semantic point of view.

The current text is a development of ideas found in [Kracht, 2003]. Since then I have spent considerable energy in getting a clearer idea on the central notion of this book, namely compositionality. In the meantime, new articles and books have appeared (for example [Barker and Jacobson, 2007]) showing that the topic is still a lively issue. I have had the benefit of extended discussions with Damir Ćavar, Lawrence Cheung, Herbert Enderton, Kit Fine, Hans-Martin Gärtner, Ben George, Fritz Hamm, László Kálmán, Ed Keenan, Ben Keil, István Kenesei, Udo Klein, Greg Kobele, András Kornai, Uwe Mönnich, Yannis Moschovakis, Chris Piñon, Nathaniel Porter, Richard Schröder and Ed Stabler. Special thanks also to István Kenesei for his support and to Damir for organising the summer school in Zadar, which got me started on this manuscript. 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.

**A Note on Notation.** This text contains lots of examples and occasional “intermissions”. The end of an example or an intermission is marked by ☼.

# Contents

<b>1</b>	<b>Synopsis</b>	<b>9</b>
<b>2</b>	<b>String Languages</b>	<b>19</b>
2.1	Languages and Grammars . . . . .	19
2.2	Parts and Substitution . . . . .	33
2.3	Grammars and String Categories . . . . .	43
2.4	Indeterminacy and Adjunction . . . . .	57
2.5	Syntactic Structure . . . . .	63
2.6	The Principle of Preservation . . . . .	71
<b>3</b>	<b>Compositionality</b>	<b>79</b>
3.1	Compositionality . . . . .	79
3.2	Interpreted Languages and Grammars . . . . .	86
3.3	Compositionality and Independence . . . . .	93
3.4	Categories . . . . .	107
3.5	Weak and Strong Generative Capacity . . . . .	115
3.6	Indeterminacy in Interpreted Grammars . . . . .	130
3.7	Abstraction . . . . .	140

<b>4</b>	<b>Meanings</b>	<b>151</b>
4.1	'Desyntactified' Meanings . . . . .	151
4.2	Predicate Logic . . . . .	159
4.3	Concepts . . . . .	165
4.4	Linking Aspects and Constructional Meanings . . . . .	174
4.5	Concepts and Pictures . . . . .	180
4.6	Ambiguity and Identity . . . . .	187
4.7	Profiling . . . . .	194
<b>5</b>	<b>Examples</b>	<b>205</b>
5.1	Predicate Logic . . . . .	205
5.2	Concept Based Predicate Logic . . . . .	212
5.3	A Fragment of English . . . . .	223
5.4	Concepts and LF . . . . .	228
5.5	The Structure of Dutch . . . . .	234
5.6	Arguing for Syntactic Structure . . . . .	245
<b>6</b>	<b>Conclusion</b>	<b>251</b>
<b>A</b>	<b>Useful Mathematical Concepts and Notation</b>	<b>253</b>
<b>B</b>	<b>Symbols</b>	<b>258</b>
<b>C</b>	<b>Index</b>	<b>260</b>
	<b>Bibliography</b>	<b>264</b>



# Chapter 1

## Synopsis

BEFORE I start with the technical discussion it is perhaps worthwhile to discuss the relevance of the concepts. I shall begin with some notes on the historical context and the current developments before I turn to the questions that I have tried to answer in this book.

Modern linguistics begins with de Saussure, yet he wrote surprisingly little on the subject matter. The famous *Cours de linguistique générale* exists in several editions none of which were published by de Saussure himself. Some years ago, however, a bundle of autographs have been found in his home in Geneva which are, I think, of supreme importance. We see de Saussure agonize over some quite basic and seemingly innocent problems: one is the distinction between what he calls “parole”, a continuous object of changing and elusive nature, and “langue”, a system of oppositions, in other words a structured object. De Saussure constantly reminds us that all the objects we like to talk about in linguistics are abstractions: meanings, letters, phonemes, and so on. The second problem that he deals with, and one that will be central to this book, is that language is a *relation* between form and meaning and not just a system of well-formed expressions.

One might think that hundred years later we have settled these issues and found satisfactory answers to them. I think otherwise. Both of the problems are to this day unsolved. To understand why this is so it is perhaps useful to look at Chomskyan linguistics. The basic ingredients of generative grammar are a firm commitment to discrete objects and the primacy of form over meaning. There is no room for gradience (though occasional attempts have been made even

by Chomsky himself to change this). Grammars are rule systems. Moreover, linguistics is for the most part the study of form, be it phonology, morphology or syntax. The rise of Montague Grammar has changed that to some degree but not entirely. One reason for this is that Montague Grammar itself, like generative grammar, is rooted in metamathematics, which puts the calculus, the mindless symbolic game, into the center of investigation.

The present book took its beginning in the realisation that what linguists (and logicians alike) call meaning is but a corrupted version thereof. A second, related insight was that linguists rarely if ever think of language as a *relation*. The ambition of the present monograph is to change that. What I shall outline here is a theory of formal languages that are not merely collections of syntactic objects but are relations between syntactic objects and their meanings.

Throughout this book, *language* means a set of signs. Signs are pairs consisting of syntactic objects and meanings. Languages are sets of signs, and hence relations between syntactic objects and meanings.

This calls for a complete revision of the terminology and the formal framework. Consider by way of example the syntactic rule

$$(1.1) \quad S \rightarrow NP VP$$

This rule can be used to replace the string /S/ by the string /NP VP/. (I use slashes to enclose strings so as to make them more visible against the text.) Yet, if language consists of syntactic objects together with their meanings we must ask what the meaning of /S/ is, or, for that matter, of /NP VP/. If anything, the meaning of /S/ is the disjunction of all possible meanings of sentences of the language, or some such object. However, notice that /S/ is not an object of any language. The whole point of auxiliary symbols in the grammar is that they are not *meant* to be part of the language for which they are used. And if they are not in the language then they have no meaning, for a language by definition endows only its own objects with meaning.

Notice that the problem existed already at the inception of grammar as production rules. Grammars never generated only the language they were designed to generate but a host of strings that did not belong to the language. Again this was precisely because they contained auxiliary symbols. While it was unproblematic if only string generation was concerned, the problem becomes more urgent if

meanings are considered as well. For now we need to replace the rule by something that replaces not only strings but signs, like this:

$$(1.2) \quad \langle S, x \rangle \rightarrow \langle NP, y \rangle \langle VP, z \rangle$$

This means something like this: an /S/ that means  $x$  can be decomposed into an /NP/ that means  $y$  and a /VP/ that means  $z$ . This formulation however is unsatisfactory. First, we have lost the idea that /S/ is replaced by the *sequence* of /NP/ followed by /VP/, for we needed to annotate, as it were, the parts by meaning. Second, there is no unique way to derive  $y$  and  $z$  from  $x$ ; rather,  $x$  is unique once  $y$  and  $z$  are given. In Montague Semantics, following Frege,  $z$  is a function, and  $x = z(y)$ , the result of applying  $z$  to  $y$ . Thus, it is actually more natural to read the rule from right to left. In that formulation it would read as follows: given an object  $\alpha$  of category NP and meaning  $y$  and an object  $\beta$  of category VP and meaning  $z$ , the concatenation  $\alpha\hat{\ } \beta$  is an object of category S and meaning  $z(y)$ . The objects can be anything; however, I prefer to use *strings*. Notice now that we have variables for strings and that we have (de facto) eliminated the syntactic categories. The rule looks more like this now:

$$(1.3) \quad \langle \alpha, y \rangle, \langle \beta, z \rangle \rightarrow \langle \alpha\hat{\ } \beta, z(y) \rangle$$

There is a proviso:  $\alpha$  must be of category NP,  $\beta$  of category VP. To implement this we say that there is a function  $f$  that takes two signs and returns a sign as follows.

$$(1.4) \quad f(\langle \alpha, y \rangle, \langle \beta, z \rangle) := \begin{cases} \langle \alpha\hat{\ } \beta, z(y) \rangle & \text{if } \alpha \text{ is of category NP,} \\ & \text{and } \beta \text{ of category VP;} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

This is the formulation that we find in Categorical Grammar and variants thereof. It is, as I see it, the only plausible way to read the rules of grammar. In this formulation the category is not explicit, as we are generating objects of the language intrinsically. The fact that the generated string  $\alpha\hat{\ } \beta$  is an S is therefore something that we must be able to recover from the sign itself. Notice that this problem exists also with the input: how do we know whether  $\alpha$  is a string of category NP? Where does this knowledge reside if not in the grammar? I shall answer some of these questions below. They show surprising complexity, and contrary to popular opinion it is not necessary to openly classify strings into categories.

From this moment on we are faced with plenty of choices. The binary function  $f$  takes as its input two signs, each of which consists of two parts. Thus it has in total four inputs. The question is whether the function is decomposable into simpler functions. Some people would argue that this is not so, and some theories encode that dictum in one or another form. Yet, from a theoretical point of view it is not good to drop a plausible hypothesis unless one really has to. The plausible hypothesis is this.

**Independence.** The functions of the grammar that create signs create the components of the signs independently of each other.

This thesis has two parts. One is the so called *Autonomy of Syntax Thesis* and the other the *Compositionality Thesis*. For convenience I spell this out for our example. The autonomy thesis says that whatever  $f(\langle\alpha, x\rangle, \langle\beta, y\rangle)$  may be in a given language, the form (or morphology) of the sign is a function of  $\alpha$  and  $\beta$  alone, disregarding  $x$  and  $y$ . The compositionality thesis says that whatever  $f(\langle\alpha, x\rangle, \langle\beta, y\rangle)$  may be in a language, its semantics depends only on  $x$  and  $y$  and nothing else. Thus we have functions  $f^\gamma$  and  $f^\mu$  such that

$$(1.5) \quad f(\langle\alpha, x\rangle, \langle\beta, y\rangle) = \langle f^\gamma(\alpha, \beta), f^\mu(x, y) \rangle$$

Translated this says that whatever form the expression takes does not depend on the meaning of the component expressions; and whatever meaning the expression has does not depend on the form of the component expressions.

What does this Principle of Independence actually say? It is at this point where many linguists start to be very creative. Anything goes in order to prove language to be compositional. But the problem is that there is little room for interpretation. Language is a relation  $R$  between expressions and meanings. What we postulate in the case of  $f$  is that there is a pair of binary functions  $f^\gamma : E \times E \rightarrow E$  and  $f^\mu : M \times M \rightarrow M$  such that (1.5). What is important is that the input signs are taken from the language  $R$  and the output sign must be in  $R$  too. Thus, independence means that we have a set of functions that generate  $R$  from the lexicon.

All functions are allowed to be partial. Partiality is essential in the generation of the signs. For example, let us see how to account for the fact that it is grammatical to say “Jack drove the car.” but not “Jack drove the bicycle.”. Clearly, we must say that “drive” requires a certain kind of vehicle. The nature of the restriction may now be either morphological or semantic. If it is morphological then it may

be formulated as a restriction on the function  $f^\gamma$  on the expressions. If however it is semantic, what to do? There are various options. The best is probably to say that the type of vehicle is already implied by the expression and so we cannot use a different one on pain of contradiction. If one dislikes this solution, here is another one. Create two modes,  $f_1$ , and  $f_2$ , and declare that  $f_1^\mu(x, y)$  is defined only if  $y$  is a motorized (earth bound) vehicle, while  $f_2^\mu(x, y)$  is defined in cases  $y$  is a different kind of vehicle. What we cannot do, however, is add some material in the syntactic structure that replicates the semantic properties, such as `carmotorized` and `bicycle-motorized`. This is effectively what has been proposed with  $\theta$ -roles. More often than not they have been used to encode semantic properties in syntax. The converse has also often been done: encode a syntactic restriction in semantics.

There is a lot of terminological ground to be covered here. If the formation of signs is a partial operation the question is whether we can at all distinguish syntactic from semantic deviance. Chomsky has argued that we can, and I wish to basically agree with his observation even though it does seem to me that it often requires some education to disentangle ungrammaticality and semantic oddness. If it is therefore possible to distinguish semantic from syntactic oddness, what could be the source of that distinction? It would be this: a sentence is syntactically well-formed if it could be generated if we looked only at the syntactic composition functions, and semantically well-formed if its meaning could be generated if we looked only at the semantic composition functions. Thus, the fact that we can distinguish between these two notions of (un)acceptability requires that we have independent knowledge of both the syntactic functions and the semantic functions. However, notice that the definition I gave is somewhat strange: how can we know the meaning of an ungrammatical sentence? What is the meaning that it has despite the fact that it is ungrammatical? Unfortunately, I do not have an answer to this question, but it is these kinds of questions that come to the fore once we make a distinction between different kinds of well-formedness. Another problem is how it is that we can at all attribute a meaning to an ungrammatical sentence. Why is it that sometimes the semantic functions are more general than the syntactic functions and sometimes the syntactic functions more general than the semantic functions? This is not only a theoretical problem. It is important also in language learning: if a child hears only correct input, it will hear sentences that are both grammatical and meaningful, so it can never (at least in principle) learn to distinguish these concepts. Again I have not much to say except noticing the problems. Part of it is that I am not concerned with learning. Another is that—surprisingly—setting up something as simple as a formal theory of inter-

preted languages as opposed to a formal theory of string languages requires much more care in the definitions, and this task has to come first. For despite the fact that the language is given in a relational form it is not clear how we can or should define from that a grammar that manipulates syntax and semantics independently. Parts of Chapter 3 are consumed by disentangling various notions of autonomy and compositionality.

Now as much as one would agree with my insistence that the language  $R$  is given a priori and cannot be adapted later, there is still a problem. Namely, no one knows for sure exactly how  $R$  looks like. This is not only due to the somewhat insufficient knowledge of what is a grammatical constituent. It has to do more with the problem of knowing exactly what the meaning of a given expression actually is. For example, what is the meaning of “drive”? Is it a function, an event, an algorithm? Is it extensional, intensional, time dependent? My own stance here is that basically expressions have propositional content, and the meaning of a proposition is its truth conditions. This implies that it is not a function in the sense of Frege (from individuals to truth values), and that the dependencies it displays result from the conditions that it places on the model. Yet, what exactly the formal nature of truth conditions is is far from clear. Logicians have unfortunately also been quite complacent in thinking that the calculi they have formulated are compositional. They mostly are not. For this reason I have to take a fresh start and develop something of a calculus of truth conditions. The problem is that certain vital constructs in logic must be discarded when dealing with natural language semantics. One of them are variables, another is type theory. To see why this is so we must simply ask ourselves what the semantics of a variable, say, “ $x$ ” is and how it differs from the semantics of a different variable, say “ $y$ ”. Moreover, these meanings should be given *independently of the form of the expression*. The result is that there is nothing that can distinguish the meaning of “ $x$ ” from that of “ $y$ ” because all there is to the difference is the difference in name. Consequently, if names are irrelevant, the meaning of the expression “ $R(x,y)$ ” is the same as “ $R(y,x)$ ”, that is, we cannot even distinguish a relation and its converse!

This observation has far reaching consequences. For if we accept that we cannot explicate same or different reference in terms of variables then the composition of meanings is severely restricted. Indeed, I shall show that it amounts to the restriction of predicate logic to some finite variable fragment. On the other hand, I will argue that nevertheless this is precisely what we want. Consider an ergative language like Dyrbal. Dixon in his [Dixon, 1994] translates the verbs of

Dyirbal by their passives in English. So, the verb meaning “hit” is translated by “is hit by”. This makes a lot of sense in Dyirbal, as it also turns out that the transitive object in Dyirbal is the syntactic pivot in coordination. Yet, we may wonder how come that “hit” can at all *mean the same thing as* “is hit by”, for “John hits Rover.” does not mean the same as “John is hit by Rover.”. The answer lies here in a distinction between meaning and meaning composition. The way the verb “hit” composes with a subject expression is certainly different from that of “is hit by”. And yet, both mean that someone hit someone.

Similarly, the issue of types is a difficult one. Take once again the meaning of the transitive verb “hit”. Montague gave it the type  $e \rightarrow (e \rightarrow t)$  (it is enough to look at the extensional type). This means that it is a function which, when given an object, returns an intransitive verb, which in turn is a function that returns a truth value when given an object. So the first object supplied is the direct object. We could think however that it is just the other way around (compare Dyirbal for that matter): the first to be supplied is the subject and the direct object comes next. Alternatively we may give it the type  $e \bullet e \rightarrow t$ , in which it gives a truth value when given a subject paired with an object. Now which of the three is correct? The problem is that they are all equivalent: choose one, get the others for free. From a technical viewpoint this is optimal, yet from our viewpoint this says that there is no a priori way to choose the types. However, from a philosophical point of view this gives rise to what has been termed Benaceraff’s Dilemma after [Benaceraff, 1973]: if we cannot choose between these formalisations how can we know that any of them is correct? That is, if there are such objects as meanings but they are abstract then how can we obtain knowledge of them? If we are serious about meanings then either we must assume that they are real (not abstract) or else that they do not exist. In particular, the idea that types are abstract properties of objects is just an illusion, a myth. Types are introduced too smoothen the relationship between syntax and semantics. They are useful but not motivated from semantics. In this connection it is important to realise that by semantics I do *not* mean model theoretic semantics. If I did, then any type assignment could be motivated from a needed fit with a particular formal model. Instead, I think of semantics primarily as truth conditions in the world.

In order to understand how this affects thinking semantically, take the sentence “John is hitting Rover.”. How can we judge whether this sentence is true? Obviously, it is of no help to say that we have to look whether or not the pair consisting of John and Rover is in the hit-relation. For it is the latter that we have to con-

struct. That we somehow possess a list of pairs where we can look the facts up is no serious suggestion. Obviously, such a list if it ever exists has to be compiled from the facts out there. But how? Imagine we are witnessing some incident between John and Rover or watching a film—where is that relation and how are we to find it? Clearly, there must be other criteria to tell us who is subject (or first argument) and who is object (or second argument). So, for a given situation we can effectively decide which object can fill the first slot and which one the second slot so that they come out as a pair in the hit-relation. Once we have established these criteria, however, there is no need to appeal to pairs anymore. For whatever it is that allows us to judge who will be subject, it is this procedure that we make use of when inserting the subject into the construction, but not earlier. The pair has become redundant. Similarly we can deal with the verbs as functions meaning, eliminating the functional nature.

A type theorist will object and say: so you are in effect changing the nature of meanings. Now they are functions from scenes (or films) to objects or whatever, but still you uphold type distinctions and so you are not eliminating types. I actually agree with this criticism. It is not types as such that I wish to eliminate. There are occasions when types are necessary or essentially equivalent to whatever else we might put in their place. What I contest is the view that the types tell us anything of essence on the syntax of the expressions. We can of course imagine languages where the fit is perfect (some computer languages are of that sort) but truth is that natural languages are definitely *not* of that kind.

I have said above that language is a relation, that is, a set of pairs. This relation is many-to-many. A given meaning can be expressed in many ways, a given expression may have many meanings. However, one may attempt to reduce the complexity by a suitable reformulation. For example, we may think that an expression denotes not several meanings but rather a single one, say, the set of all its meanings. Call this kind of meaning *set-meaning* and the other the *ground meaning*. Thus, /crane/ denotes a set of ground meanings, one covering the bird meaning and another the machine meaning. This technical move eliminates polysemy and makes language a function from expressions to (set-)meanings. There are however many problems with this approach. The first is that the combination of two set-meanings is much more complex than the combination of ground meanings, for it must now proceed through a number of cases. Consider namely how complex signs are being made. Given a two place function  $f$ , a complex sign is made from two simple signs, each being an expression paired with a ground meaning. It



is thus particular expressions with particular ground meanings that are composed via  $f$ , and not expressions with all their meanings or meanings with the totality of their expressions. If an expression is polysemous the claim is therefore that it must enter with any one of its meanings in place of the collection of all its meanings. The expression /big crane/ can therefore be formed with two particular meanings for /crane/, each of them however taken on its own. The expression is thus again polysemous insofar as the combination of “big” with any of the two ground meanings makes sense. Similarly, /all cranes/ can never be a quantification over objects of the expression /cranes/ in both senses simultaneously. It can only be either of them: a quantification over some birds, or a quantification over some machines. Lumping the two meanings into a set therefore creates options that languages do not seem to have. Or, more precisely, the fact that a given expression has two ground meanings (= is polysemous) is technically different from it having a set-meaning.

As the reader will no doubt notice the present monograph is quite technical. This is because I felt it necessary to explore certain technical options that the setup leaves us with. Since the details are essentially technical there is no point in pretending that they can be dealt with in an informal way. Moreover, if we want to know what the options are we better know as exactly as possible what they consist in. It so turns out that we can obtain certain results on the limitations of compositionality. Moreover, I show that certain technical manoeuvres (such as introducing categories or eliminating polysemy) each have nontrivial side effects that need to be addressed. By doing this I hope to provide the theoretical linguist with a tool for choosing among a bewildering array of options.



# Chapter 2

## String Languages

THIS chapter introduces the notion of a grammar as an algebra. 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*.

### 2.1 Languages and Grammars

Languages in the way they appear to us seem to consist of strings. The text in front of you is an example. It is basically a long chain of symbols, put one after the other. Yet, linguists stress over and over again that there is *structure* in this chain, and that this structure comes from a *grammar* that generates this language. I shall assume that the reader is familiar with this standard view on language. In this chapter I shall rehearse some of the definitions, though taking a slightly different view. While standard syntactic textbooks write rules in the form of replacement rules ( $S \rightarrow NP VP$ ) to be thought of as replacing what is to the left by what is to the right, here we take a bottom up view: we define grammars as devices that *combine* expressions. The reasons for this shift have already been discussed. This is also the way in which Montague defined his formation rules.

Although I shall have very little to say about phonology I should make it clear that when I use the terms “alphabet” and “letter” you may replace them by “phoneme inventory” and “phoneme”. Likewise, we may decide to include tone and other characteristics into the representation. All this can be done. The only reason that I do not do it is that it would distract the attention from the central issues. The reader is however asked to keep in mind that the discussion is largely independent of the actual nature and manifestation of the alphabet.

I said that languages are sets of strings. Clearly, there is more to languages, as they also give meanings to the strings. Yet, if we disregard this latter aspect—and maybe some more—, we retain as the simplest of all manifestations of a language: that of a *set of strings*. The topic of string languages is very rich since it has been thoroughly studied in formal language theory. We start therefore by discussing string languages.

Recall that a **string** over some alphabet  $A$  is a sequence of letters from  $A$ ; for example,  $/\text{abc}b\text{ab}/$  is a string over  $\{a, b, c\}$ . It is also a string over the alphabet  $\{a, b, c, d\}$  but not over  $\{a, b\}$ . Alternatively, a string over  $A$  is a function  $\vec{x} : n \rightarrow A$  for some natural number  $n$  (see Appendix);  $n$  is the **length** of  $\vec{x}$ . If  $n = 0$  we get the **empty string**; it is denoted by  $\varepsilon$ . We write  $\vec{x}, \vec{y}$  (with an arrow) for arbitrary strings. Concatenation is either denoted by  $\vec{x}\vec{y}$  or by juxtaposition. In running text, to enhance explicitness, I enclose material strings (or exponents in general) in slashes, like this:  $/\text{dog}/$ . This carries no theoretical commitment of any sort.

**Definition 2.1** *Let  $A$  be a finite set, the so-called **alphabet**.  $A^*$  denotes the set of strings over  $A$ ,  $A^+$  the set of nonempty strings. A **language over  $A$**  is a subset of  $A^*$ .*

Following Unix convention, we shall enclose names for sets of symbols by colons (for example,  $:\text{digit}:$ ). This way they cannot be confused with sets of strings, for which we use ordinary notation.

**Definition 2.2** *The union of two sets is alternatively denoted by  $S \cup T$  and  $S \mid T$ . Given two sets  $S$  and  $T$  we write*

$$(2.1) \quad S \cdot T := \{\vec{x}\vec{y} : \vec{x} \in S, \vec{y} \in T\}$$

*Furthermore,  $S^n$  is defined inductively by*

$$(2.2) \quad S^0 := \{\varepsilon\}, \quad S^{n+1} := S^n \cdot S$$

Finally we put

$$(2.3) \quad S^* := \bigcup_{n \in \mathbb{N}} S^n$$

as well as

$$(2.4) \quad S^+ := S \cdot S^*$$

Typically, we write  $ST$  in place of  $S \cdot T$ .  $*$  binds stronger than  $\cdot$ , and  $\cdot$  binds stronger than  $\cup$ . We shall write  $\{x\}$  and  $x$  indiscriminately, when  $x$  is a single letter.

It is important to note that a language as defined here is a *set*, so it is unstructured. 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. Thus, for now a grammar is a system of rules (or rather functions). It is the grammar that imposes structure on a language. This point seems contentious; in fact, many linguists think differently. They think that the language itself possesses a structure that needs to be described using the grammar. Some are convinced that some descriptions (maybe even a single one) is better than all the others (see [Tomalin, 2006] on the origin of this view). I consider this belief unfounded. That we know the right grammar when we see it is wishful thinking. It is clear that regularities need accounting for. However, that accounting for them in the right way will make the rule apparatus more transparent needs to be demonstrated. The most blatant defect of such claims is that no one knows how to define simplicity in an unambiguous way. One exception is perhaps Kolmogorov complexity, which is however difficult to use in practice (see [Kornai, 2007] on that subject). In absence of a unique notion of simplicity we are left with the intuition that a language “calls” for a particular description in the form of a certain grammar. But it may well be that there are different descriptions of the same facts, none of which need to be essentially better than the other. Indeed, if one looks around and studies various frameworks and the way they like to deal with various phenomena, one finds that there is little fundamental consensus; nor is there a criterion by which to judge who is right. Thus, a language may possess various quite different grammars. These grammars in turn impose different structures on the language and it may be impossible to say which one is “correct”. Thus the distinction must be made between the set of acceptable strings and the structure that we see in them.

**Example 1.** (See also Example 6 below.) The language of *unbracketed* additive arithmetical terms (or *ua-terms* for short) is defined as follows. Consider the set

$$(2.5) \quad \text{:digit:} := \{0, 1, \dots, 9\}$$

An **ua-term** is a string over this alphabet plus the additional symbol  $/+/\mathit{}$  such that it neither ends nor begins with  $/+/\mathit{}$ . So it is a member of the following set:

$$(2.6) \quad \text{UA} := \text{:digit:}^+ (\text{+:digit:}^+)^*$$

Examples are

$$(2.7) \quad 0, 10, 010+7, 00+01+31, 1001+000+9$$

In practice we think of such a string as consisting of blocks of digits separated by  $/+/\mathit{}$ . This is so far just a matter of convenience. We shall see below however why this view is justified.

In contrast to the unbracketed arithmetical terms, the bracketed arithmetical terms (**a-terms**) always have brackets. They are technically strings over a different alphabet, namely  $\text{:digit:} \cup \{+, (, )\}$ . Thus, it is not that we do not write ua-terms with brackets; they do not contain any brackets in the first place. An a-term, by contrast, has them everywhere. (A precise definition of a-terms will be given in Example 6.) There are many ways to ‘analyse’ a given ua-term as arising from some a-term. For example, we can think of the ua-term

$$(2.8) \quad \vec{x}_0 + \vec{x}_1 + \vec{x}_2 + \dots + \vec{x}_n$$

as being derived in a left bracketed (2.9) or right bracketed (2.10) way:

$$(2.9) \quad (\vec{x}_0 + (\vec{x}_1 + (\vec{x}_2 + \dots (\vec{x}_{n-1} + \vec{x}_n) \dots)))$$

$$(2.10) \quad (((\dots ((\vec{x}_0 + \vec{x}_1) + \vec{x}_2) + \dots \vec{x}_{n-1}) + \vec{x}_n)$$

Similarly, the ua-term

$$(2.11) \quad 3+1+7+5$$

can be derived from the following a-terms by deleting brackets:

$$(2.12) \quad (((3+1)+7)+5), ((3+(1+7))+5), (3+((1+7)+5)), (3+(1+(7+5))).$$

There is no way to decide which analysis is correct.



**Example 2.** The formation of the third singular present of the English verb is identical to the plural of nouns. It consists—irregular forms and idiosyncrasies of spelling aside—in the addition of an /s/, /es/ or /ses/. Is there a formal identity between the two or do they just accidentally happen to be the same? 🌐

Let me briefly go into some details. Ordinary languages contain—apart from the obvious alphabetic characters—also punctuation marks; in addition to punctuation marks we find the digits, and the blank, written here /\_/ throughout when quoting material language strings, and, finally, some less obvious characters such as “newline” or “new paragraph”. These should be counted into the alphabet *A* for the purposes of writing serious grammars for languages. There is, for example, a difference in English between /black\_ bird/ and /blackbird/. In written English the only difference is the presence or absence of the blank; in spoken English this comes out as a different stress assignment. The same goes obviously for punctuation (the difference between restrictive and nonrestrictive relative clauses is signalled by the presence of a comma). Spoken language has intonation, which is absent from written language; punctuation is a partial substitute for intonation. In what is to follow, we will concentrate on written language to avoid having to deal with issues that are irrelevant for the purpose of this book. Writing system however introduce their own problems. For matters concerning the intricacies of alphabets I refer the reader to [Korpela, 2006].

**Intermission 1.** Some interesting facts about punctuation. In general, there is something of a syntax of punctuation marks. Writing no blank is different from writing one blank, while one blank is the same as two (consecutive) blanks. Two periods are likewise the same as one, two commas the same as one, and so on. In general, punctuation marks act as separators, not as brackets. Separators come in different strengths. For example, a period is a stronger separator than a comma. This means that if a period and a comma will be in competition, the (sentence) period will win. 🌐

Anyone who is nowadays dealing with characters will know that there is a lot of structure in an alphabet, much the same way as the set of phonemes of a language is highly structured. There is first and foremost a division into alphabetic characters, digits, and punctuation marks. However, there is an additional division into such characters that serve as separators and those that do not. Separators are there to define the units (“identifiers” or “words”). For ua-terms, /+/ is a separator. Separators could also be strings, of course. If we want to understand where the

words are in a text we break a string at all those positions where we find a separator. Thus, the blank and also punctuation marks are typical separators. But this is not always the case. A hyphen, for example, is a punctuation mark but does not serve as a separator—or at least not always. In programming languages, brackets are separators; this means that the name of a variable may not contain brackets, since they would simply not be recognised as parts of the name. Anyone interested in these questions may consult, for example, books or manuals on regular expressions and search patterns.

While we often think of languages as being sets of strings over a given alphabet, there are occasions when we prefer to think of languages as somehow independent of the alphabet. These viewpoints are not easy to reconcile. We can introduce some abstractness as follows. Let  $A$  and  $B$  be alphabets and  $m : A \rightarrow B^*$  a map.  $m$  induces a **homomorphism**  $\bar{m} : A^* \rightarrow B^*$  in the following way.

$$(2.13) \quad \bar{m}(x_0x_1 \cdots x_{n-1}) := m(x_0)\hat{\ }m(x_1)\hat{\ } \cdots \hat{\ }m(x_{n-1})$$

Then  $\bar{m}[L]$  is the **realphabetisation** of  $L$ .

**Example 3.** In German, Umlaut refers to the change of /a/, /o/ and /u/ to /ä/, /ö/ and /ü/, respectively. Standard German allows to replace the vowels with dots by a combination of the vowel with /e/ (historically, this is where the dots come from; they are the remnants of an /e/ written above the vowel). So, we have a map  $m : a \mapsto ae, \ddot{o} \mapsto oe, \ddot{u} \mapsto ue$ . For all other (small) letters,  $m(x) = x$ . Hence,

$$(2.14) \quad \bar{m}(\text{Rädelsführer}) = \text{Raedelsfuehrer}$$



We then say that we look at a language only up to realphabetisation. In linguistics this is done by considering spoken language as primary and all written languages as realphabetisations thereof. Usually we will want to require that  $\bar{m}$  is injective on  $L$ , but spelling reforms are not always like that. In Switzerland, the letter /ß/ is written /ss/, and this obliterates the contrast between /Maße/ ‘measures’ and /Masse/ ‘mass’. For this reason we shall not deal with realphabetisation except for theoretical purposes, where we do require that  $\bar{m}$  be injective. Realphabetisations are not structurally innocent. What is segmentable in one alphabet may not be in another. Imagine an alphabet where /downtown/ is rendered by a single letter, say, / $\blacktriangle$ / . The map sending / $\blacktriangle$ / to /downtown/ makes an indecomposable unit



decomposable (/down/ + /town/). The dependency of the analysis on the alphabet is mostly left implicit throughout this work.

The division into units, which are so important in practical applications (witness the now popular art of tokenisation), is from a theoretical standpoint secondary. That is to say, it is part of the responsibility of a grammar to tell us what the units are and how to find them. Whether or not a symbol is a separator will be a consequence of the way the grammar works, not primarily of the language itself. This is why we may maintain, at least in the beginning, that the alphabet too is an unstructured set. The structure that we see in the language and its alphabet is—as I emphasised above—imposed on it through a system of rules and descriptions, in other words a *grammar*. This applies of course to phonemes and features in the same way.

In my view, a grammar is basically an interpretation of an abstract language. In computer science one often talks about **abstract** and **concrete syntax**. The abstract syntax talks about the ideal constitution of the syntactic items, while the concrete syntax specifies how the items are communicated. The terminology used here is that of “signature” (abstract) versus “grammar” (concrete).

**Definition 2.3** *Let  $F$  be a set, the set of function symbols. 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,  $f : S^0 \hookrightarrow S$ , but there is little gain in allowing a zeroary function to be partial. 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 2.4** *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.  $\mathcal{J}$  is*

called the **interpretation**. If  $\Omega(f) = 0$ ,  $f$  is called **lexical**, otherwise **nonlexical**. The set  $\{\mathcal{J}(f) : \Omega(f) = 0\}$  is called the **lexicon** of  $G$ , and the set  $\{\mathcal{J}(f) : \Omega(f) > 0\}$  the set of **rules**. The **language** generated by  $G$ , in symbols  $L(G)$ , is defined to be the least set  $S$  satisfying for every  $f \in F$  and all  $\vec{x}_i \in A^*$ ,  $i < \Omega(f)$ :

$$(2.15) \quad \text{If for all } i < \Omega(f) : \vec{x}_i \in S \text{ then } \mathcal{J}(f)(\vec{x}_0, \dots, \vec{x}_{\Omega(f)-1}) \in S$$

**Example 4.** Let  $F := \{j, t, f\}$ , and  $\Omega(j) = \Omega(t) = 0$ ,  $\Omega(f) = 2$ . Now,  $\mathcal{J}$  is defined as follows.  $\mathcal{J}(j)$  is a zeroary function, and so  $\mathcal{J}(j)()$  is a string, the string `/John/`. Likewise,  $\mathcal{J}(t)() = \text{talks}$ . Finally, we look at  $\mathcal{J}(f)$ . Suppose first that  $\mathcal{J}(f)$  is interpreted like this.

$$(2.16) \quad \mathcal{J}(f)(\vec{x}, \vec{y}) := \vec{x} \frown \vec{y}.$$

Then the language contains string like this one:

$$(2.17) \quad \text{John\_talks.\_talks.}$$

The function  $\mathcal{J}(f)$  needs to be constrained. One obvious way is to restrict the first input to `/John/` and the second to `/talks/`. An indirect way to achieve the same is this definition.

$$(2.18) \quad \mathcal{J}(f)(\vec{x}, \vec{y}) := \begin{cases} \vec{x} \frown \vec{y} & \text{if } \vec{x} \text{ ends with /n/} \\ & \text{and } \vec{y} \text{ begins with /t/;} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

This grammar has the following language:

$$(2.19) \quad \{\text{John, talks, John\_talks.}\}$$



**Example 5.** Here is now a pathological example. A set  $S$  is called **countable** if it is infinite and there is an onto function  $f : \mathbb{N} \rightarrow S$ . If  $S$  is countable we can assume that  $f$  is actually bijective. Let  $L \subseteq A^*$ .  $L$  is countable, since  $A$  is finite. Let  $f : \mathbb{N} \rightarrow L$  be bijective. Let now  $F := \{b, s\}$ ,  $\Omega(b) := 0$ , and  $\Omega(s) := 1$ . This means that we get the following terms:  $b, s(b), s(s(b)), s(s(s(b))), \dots$  The general

element has the form  $s^n(b)$ ,  $n \in \mathbb{N}$ . This is a familiar way to generate the natural numbers: start with zero and keep forming successors. Further, we put

$$(2.20) \quad \begin{aligned} \mathcal{J}(b)() &:= f(0) \\ \mathcal{J}(s)(\vec{x}) &:= f(f^{-1}(\vec{x}) + 1) \end{aligned}$$

So, we start with the first element in the enumeration  $f$ . If given  $\vec{x}$  its number in the enumeration ( $f^{-1}(\vec{x})$ ). If we add 1 to this number and translate this via  $f$  we get the next element in the list. In other words, we have  $\mathcal{J}(s)(f(n)) = f(n + 1)$ .

This grammar generates  $L$ . It follows that every countable language has a grammar that generates it.  $\odot$

Evidently, any  $f \in F$  (that is, every mode) is either lexical or nonlexical. Notice that there are no requirements on the functions, not even that they be computable. (Recently, [Lasersohn, 2006] has argued that computability may not even be an appropriate requirement for meanings. Without endorsing the argument that he presents I have dropped the requirement here.) We shall introduce restrictions on the functions as we go along. The lexicon is not always considered part of the grammar. I make no principled decision here; it is just easier not to have to worry about the rules and the lexicon separately.

**Example 6.** This is one of our main examples: it will be called *the language of equations*.

$$(2.21) \quad \text{:eq:} := \text{:digit:} \cup \{+, -, (, ), =\}$$

$F = \{f_0, f_1, f_2, f_3, f_4, f_5, f_6\}$ .  $\Omega(f_0) = \Omega(f_1) = 0$ ,  $\Omega(f_2) = \Omega(f_3) = 1$ ,  $\Omega(f_4) = \Omega(f_5) = \Omega(f_6) = 2$ .  $\vec{x}$  is **binary** if it only contains  $/0/$  and  $/1/$ ;  $\vec{x}$  is an a-term if it does not contain  $/=/$ . The modes are shown in Table 2.1. The strings that this grammar generates are of the following form. They are either strings consisting of the letters  $/0/$  and  $/1/$ , for example  $/010/$ ,  $/11101/$ , or they are a-terms, like  $/(1+(01-101))/$ ; or they are equations between two such a-terms, like  $/(1+10)=11/$ . (A single numeral expression also is an a-term.)  $\odot$

Given a signature  $\Omega$ , we define the notion of an  $\Omega$ -term.

**Definition 2.5** Let  $V$  be a set of variables disjoint from  $F$ . Let  $\Omega$  be a signature over  $F$ . An  $\Omega$ -term over  $V$  is a string  $t$  over  $F \cup V$  satisfying one of the following.

Table 2.1: The Modes of Example 6

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

- ❶  $t \in V$ ,
- ❷  $t = f$ , where  $\Omega(f) = 0$ ,
- ❸  $t = f^{\wedge} t_0^{\wedge} \cdots^{\wedge} t_{n-1}^{\wedge}$ , where  $n = \Omega(f)$  and  $t_i$  is an  $\Omega$ -term for every  $i < n$ .

The symbol  $\text{Tm}_{\Omega}(V)$  denotes the set of all  $\Omega$ -terms over  $V$ . The set  $\text{Tm}_{\Omega}(\emptyset)$  is of special importance. It is the set of **constant  $\Omega$ -terms**. A term  $t$  is **constant** if  $t \in F^+$ , that is, if it contains no variables. Given a grammar  $G = \langle \Omega, \mathcal{J} \rangle$ , we also call an  $\Omega$ -term a  **$G$ -term**.

See Figure 2.2 on Page 53 for an example of term. Notice that the second case is a subcase of the third (where  $n = 0$ ). It is listed separately for better understanding. Some remarks are in order. Standardly, terms are considered abstract, but I thought it easier to let terms also be concrete objects, namely strings. The syntax chosen for these objects is Polish Notation. It has the advantage of using the alphabet itself and having the property of transparency (see Page 65 for a definition).

Exercises 4 and 5 show that the language enjoys unique readability. Delaying the justification for the terminology, let us make the following definition.

**Definition 2.6** *Let  $t$  be an  $\Omega$ -term.  $s$  is a **subterm** of  $t$  if and only if  $s$  is an  $\Omega$ -term and a substring of  $t$ .*

Based on the exercises at the end of this section one can show that the language of terms is quite well behaved. A substring that looks like a term actually is a subterm under every analysis. (Consequently there can be only one analysis.)

**Proposition 2.7** *Let  $s$  and  $t$  be  $\Omega$ -terms and  $s$  a substring of  $t$ . Then either  $s = t$  or  $t = f \widehat{t}_0 \cdots \widehat{t}_{n-1}$  for some  $f$  and  $n = \Omega(f)$  and there is an  $i < n$  such that  $s$  is a subterm of  $t_i$ .*

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  $n = \Omega(f)$ .

We call  $\iota_G$  the **unfolding function** and say that  $t$  **unfolds in  $G$  to  $\vec{x}$**  if  $\iota_G(t) = \vec{x}$ . If the grammar is clear from the context, we shall write  $\iota(t)$  in place of  $\iota_G(t)$ . Continuing our example, we have

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

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

$$(2.23) \quad \text{dom}(\iota) := \{t \in \text{Tm}_\Omega(\emptyset) : \iota(t) \text{ is defined}\}$$

the set of **orthographically definite terms**. The term  $f_4 f_3 f_0 f_2 f_1$  is orthographically definite, while the term  $f_6 f_6 f_0 f_1 f_1$  is not. This is because once  $f_6$  has been

used, it introduces the symbol  $/=$ , and none of the modes can apply further. If  $t$  is orthographically definite, so is any subterm of  $t$ . Notice that for a grammar  $G$ , the language can simply be defined as

$$(2.24) \quad L(G) := \{t(t) : t \in \text{Tm}_\Omega(\emptyset)\}$$

Notice that this is different from the standard concept. This difference will be of great importance later on. Standardly, grammars may contain symbols other than the terminal symbols. The nonterminal alphabet contains characters foreign to the language itself. While in formal languages the presence of such characters can be motivated from considerations of usefulness, in our context these symbols make no sense. This is because we shall later consider interpreted languages; and there is, as far as I know, no indication that the nonterminal symbols have any meaning. In fact, in the terminology of this book, by the definition of “language” and “nonterminal symbol” the latter have no meaning. All of this will follow from the principles defined in Section 2.6. The present requirement is weaker since it does not constrain the power of the rules. What it says, though, is that the generation of strings must proceed strictly by using strings of the language itself. Later we shall also require that the strings must be used in the meaning that the language assigns to them.

If we eliminate nonterminal symbols, however, a lot of things change as well.  $L(G)$  not only contains the strings at the end of a derivation but every string that is built on the way. If, for example, we write our grammar in a context free fashion,  $L(G)$  not only contains the sentences, but the individual words, and all constituents that any sentence of  $L(G)$  has. Therefore, unlike in traditional linguistic theory,  $L$  is not simply assumed to contain sentences, but all constituents. To distinguish these two notions we shall talk of **language in the narrow sense** if we mean language as a set of sentences; and we speak of **language in the wide sense**—or simply of **language**—otherwise. Notice that the difference is merely the way in which the language defines its grammar. As objects both are sets. But a language in the narrow sense leaves larger room to define grammars as languages in the narrow sense also fix the set from which constituents may be drawn. Our stance in the matter is that one should start with language in the wider sense. The reasons for this will I hope become clear in Chapter 3. At this moment I’d like to point out that for all intents and purposes starting with language in the narrow sense makes the grammar radically underdetermined.

For the working linguist, the choice of  $L$  is a highly empirical matter and hence full of problems: in defining  $L$  we need to make decisions as to what the

constituents of the language are. This means we need more input in the first place. On the other hand, we get a more direct insight into structure. A grammar can only analyse a string into parts that are already members of  $L$ . Of course there is still a question of whether a given string really occurs as a constituent (we shall discuss that point later). But it can only do so if it is in  $L$ . A side effect of this is that we can sometimes know which occurrences of symbols are syncategorematic. Basically, an occurrence of a symbol is syncategorematic in a string under a derivation if it is not part of any primitive string which the derivation uses. This is admittedly vague; a proper definition must be deferred to Section 2.6.

**Example 7.** I give two alternative formulations of Boolean logic. The alphabet is as follows.

$$(2.25) \quad \text{bool} := \{\emptyset, 1, p, \neg, \wedge, \vee, (, )\}$$

The first language is the smallest set  $S$  satisfying the equation (here, as in the sequel,  $\cdot$  binds stronger than  $|$  or  $\cup$ ):

$$(2.26) \quad S = (p \cdot (\emptyset \mid 1)^*) \cup (\cdot \neg \cdot S \cdot) \cup (\cdot S \cdot (\vee \mid \wedge) \cdot S \cdot)$$

The other language is the union  $D \cup S$ , where  $D$  and  $S$  are the minimal solution of the following set of equations:

$$(2.27) \quad \begin{aligned} D &= D \cup (\emptyset \mid 1) \cdot D \\ S &= p \cdot D \cup (\cdot \neg \cdot S \cdot) \cup (\cdot S \cdot (\vee \mid \wedge) \cdot S \cdot) \end{aligned}$$

It turns out that in both cases  $S$  is the same set; however, in the first example the language defined is just  $S$ , in the second it is  $S \cup D$ .  $S$  contains  $p\emptyset 1$ ,  $(\neg p\emptyset)$ ,  $(p1 \wedge (\neg p1))$ .  $D$  (but not  $S$ ) also contains  $\emptyset$ ,  $111$ .  $\otimes$

Given a grammar  $G$  and a string  $\vec{x}$ , we call a term  $t$  an **analysis term** or simply 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**. A *grammar* is called **ambiguous** if it generates at least one ambiguous string, and **unambiguous** otherwise.

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

**Exercise 2.** Write grammars for the unbracketed additive terms, the left and the right bracketed additive terms of Example 1, respectively.

**Exercise 3.** Terms are strings, by definition, and can therefore be looked at as members of a language. The methodology of this book can therefore also be applied to them. Consider, by way of example, the strings for terms in Example 6. Write a grammar for the set of all terms; then write a grammar for the set of all orthographically definite terms.

**Exercise 4.** The formal notation of terms must be accompanied by a proof that it is uniquely readable. We shall use this and the next exercise to deliver such a proof. Recall that terms are sequences of function symbols, no extra symbol is added. However, not every such sequence is a term. Let  $\Omega$  be a signature. For  $f \in F \cup V$  let  $\gamma(f) := \Omega(f) - 1$ , and for a string  $\vec{x} = x_0x_1 \cdots x_{n-1} \in F^*$  let  $\gamma(\vec{x}) = \sum_{i < n} \gamma(x_i)$ . Show the following: if  $\vec{x} \in F^*$  is a term, then (i)  $\gamma(\vec{x}) = -1$ , and (ii) for every proper prefix  $\vec{y} = x_0x_1 \cdots x_{m-1}$ ,  $m < n$ ,  $\gamma(\vec{y}) \geq 0$ . (It follows from this that no proper prefix of a term is a term.) *Hint.* Do induction on the length.

**Exercise 5.** (Continuing the previous exercise.) Let  $\vec{x} = x_0x_1 \cdots x_{n-1} \in F^*$  be a string. Then if  $\vec{x}$  satisfies (i) and (ii) from the previous exercise,  $\vec{x}$  is a term. *Hint.* Induction on  $n$ . The cases  $n = 0, 1$  are straightforward. Now suppose that  $n > 1$ . Then  $x = x_0x_1 \cdots x_{n-1}$  and  $\gamma(x_0) = p \geq 0$ , by (ii). Show that there is a number  $i > 1$  such that  $\gamma(x_1 \cdots x_{i-1}) = -1$ ; and we choose  $i$  minimal with that property. Hence,  $\vec{y}_0 = x_1 \cdots x_{i-1}$  is a term, by inductive assumption. If  $p > 1$  we have  $i < n$ , and there is  $i' > i$  such that  $\vec{y}_1 = x_ix_{i+1} \cdots x_{i'}$  such that  $\vec{y}_1$  is a term. And so on, getting a decomposition  $x_0\vec{y}_0 \cdots \vec{y}_p$ .

**Exercise 6.** Show Proposition 2.7. *Hint.* Assume that  $s \neq t$ . Then there is a decomposition  $t = f \hat{\ } t_0 \hat{\ } \cdots \hat{\ } t_{n-1}$ . Now fix a substring occurrence of  $s$  in  $t$ . Assume that it starts in  $t_i$ . Then show that it must also end in  $t_i$ .



## 2.2 Parts and Substitution

We defined a language to be the set of all expressions. Since we do not discriminate sentences from nonsentences the language contains not only sentences but all expressions. It therefore seems possible to say of some expressions whether one is a part of the other. For example, we would like to say that /The cat is on the mat./ contains /on the mat/ as its part, but not, for example, /cat is on/ or /dog/. In the cases just given this is straightforward: /cat is on/ is not in our language (in the wide sense), it has no meaning; /dog/ on the other hand is not a string part of the expression. In other cases, however, matters are not so easy. Is /Mary ran/ a part of /John and Mary ran./ or is it not? It is a string part of the sentence and it is meaningful. As it turns out, there is no unique answer in this case. More problems arise, making the notion of *part* quite elusive. One problem is that we have so far no conditions on the string functions; another is that a given string may have been composed in many different ways. Let us discuss these issues. We begin however with a definition of *part*.

**Definition 2.8** *Let  $G$  be a grammar.  $\vec{x}$  is a **part of**  $\vec{y}$  if there are constant terms  $s$ ,  $u$  such that  $s$  is a subterm of  $u$  and  $\iota_G(s) = \vec{x}$  and  $\iota_G(u) = \vec{y}$ .*

This definition of *part of* pays no attention to the strings. Instead it looks at the way the strings are obtained through the string functions of the grammar. Thus, any useful restriction will come from restricting the string functions. The definition also pays no attention to the way in which the parts occur in the larger string. Occurrences will be defined in Definition 2.11, and then we shall review Definition 2.8. The examples of this section will show how broad the spectrum of grammars is and how it affects parthood.

**Example 8.** Consider a unary function  $f$  which forms the past tense, for example  $\mathcal{J}(f)(\text{go}) = \text{went}$ ,  $\mathcal{J}(f)(\text{sing}) = \text{sang}$ ,  $\mathcal{J}(f)(\text{ask}) = \text{asked}$ . In this grammar, /go/ is a part of /went/, /sing/ a part of /sang/, /ask/ a part of /asked/. ☹

In standard terminology it is actually not assumed that /went/ is literally made from /go/; rather, it is assumed that the verb ‘to go’ possesses different allomorphs, and the context decides which of them is going to be used. At the end of this chapter we shall propose that syntactic functions may not delete material. This

takes care of the problem by eliminating the grammar from Example 8. Let us now look at a second example.

**Example 9.** We present two ways of generating the nonempty strings over the alphabet  $\text{blet} := \{a, b\}$  of binary letters.  $C_1$  consists of the zeroary functions  $f_a$ ,  $f_b$  plus the unary functions  $f_0$  and  $f_1$ . We have

$$(2.28) \quad \begin{aligned} \mathcal{J}_1(f_a)() &:= a \\ \mathcal{J}_1(f_b)() &:= b \\ \mathcal{J}_1(f_0)(\vec{x}) &:= \vec{x} \frown a \\ \mathcal{J}_1(f_1)(\vec{x}) &:= \vec{x} \frown b \end{aligned}$$

So,  $\iota_{C_1}(f_1 f_0 f_0 f_a) = \text{aaab}$ . This grammar is the ‘typewriter model’ of strings. Strings are generated by appending letters one by one to the initial letter.

The grammar  $C_2$  has the zeroary function symbols  $f_a$  and  $f_b$  and a binary symbol  $a$ . We have

$$(2.29) \quad \begin{aligned} \mathcal{J}_2(f_a)() &:= a \\ \mathcal{J}_2(f_b)() &:= b \\ \mathcal{J}_2(a)(\vec{x}, \vec{y}) &:= \vec{x} \vec{y} \end{aligned}$$

For example,  $\iota_{C_2}(a a f_a f_a a f_a f_b) = \text{aaab}$ .

In  $C_1$ ,  $\vec{x}$  is part of  $\vec{y}$  if and only if it is a nonempty prefix of  $\vec{y}$ . In  $C_2$ ,  $\vec{x}$  is a part of  $\vec{y}$  if and only if it is a nonempty subword.  $\odot$

It is to be noted that both grammars generate the set  $A^+$ , so they are extensionally identical. Yet structurally they are distinct. According to  $C_2$  strings can have many more parts than according to  $C_1$ . For example,  $/\text{aaab}/$  possesses (apart from itself) the parts  $/a/$ ,  $/aa/$ ,  $/aaa/$ ,  $/b/$ ,  $/ab/$ ,  $/aab/$ . In addition, the string  $/aa/$  has two occurrences in  $/\text{aaab}/$ , which we may denote as follows:  $/\underline{aa}ab/$ , and  $/aa\underline{ab}/$ . (More on occurrences in Definition 2.11 and Section 2.5.) Both occurrences are actually parts of the string. It turns out, though, that not all parts can be parts in one and the same derivation. The more useful notion is in fact defined for a particular analysis term. The relation “is part of” is then the union of the relations “is a  $t$ -part of” for terms  $t$ .

**Definition 2.9** Let  $G$  be a grammar,  $t$  a constant term and  $\vec{x}$  and  $\vec{y}$  strings. We say that  $\vec{x}$  is a ***t-part of***  $\vec{y}$  if  $\iota_G(t) = \vec{y}$  and there is a subterm  $s$  of  $t$  such that  $\iota_G(s) = \vec{x}$ . In this case there is  $t'(x)$  such that  $t = t'(s)$ .

With certain adaptations we can say that the relation “is a  $t$ -part of” is transitive. (If  $\vec{y}$  is a  $t$ -part of  $\vec{z}$  and is the unfolding of  $s$ ,  $s$  a subterm of  $t$ , then parts of  $\vec{y}$  must be  $s$ -parts of  $\vec{y}$  in order to be  $t$ -parts of  $\vec{z}$ .) Here is a somewhat surprising result given that the union of transitive relations need not be transitive.

**Proposition 2.10** *The relation is part of is transitive.*

**Proof.** Let  $\vec{x}$  be a part of  $\vec{y}$  and  $\vec{y}$  a part of  $\vec{z}$ . Then there are terms  $r$  and  $s$  such that  $r$  unfolds to  $\vec{x}$  and  $s$  unfolds to  $\vec{y}$  and  $r$  is a subterm of  $s$ . Furthermore there are  $t$  and  $u$  that unfold to  $\vec{y}$  and  $\vec{z}$ , respectively, and  $t$  is a subterm of  $u$ . Since they unfold to the same string, we may replace  $t$  in  $u$  by  $s$ , giving us a new term  $u'$ , of which  $s$  is a subterm. Since  $r$  is a subterm of  $s$ , it is also a subterm of  $u$ .  $\square$

Given a single  $C_2$ -term  $t$  for /aaab/, the substring occurrences that correspond to the subterms actually form a tree. This is essentially because the grammar encodes a context free analysis. However,  $C_2$  is ambiguous: /aaab/ has several analysis terms, and they provide different constituent analyses. The analysis terms are as follows:  $aaaf_a f_a f_a f_b$ ,  $af_a a f_a f_a f_b$ ,  $af_a f_a a f_a f_b$ , and  $af_a a f_a a f_a f_b$ . On the other hand,  $C_1$  is unambiguous.

Standard tests for constituency in textbooks include the *substitution test*. Before we look in detail at the test let us first say a few words about string substitution.

**Definition 2.11** A (**1-**)*context* is a pair  $C = \langle \vec{x}, \vec{z} \rangle$  of strings. We say that  $\vec{y}$  **occurs in**  $\vec{u}$  if  $\vec{u} = \vec{x}\vec{y}\vec{z}$ . Also,  $C$  is an **occurrence of**  $\vec{y}$  **in**  $\vec{u}$ . We write  $\vec{u} = C(\vec{y})$ . The result of substituting  $\vec{w}$  for  $\vec{y}$  in its occurrence  $C$  is  $C(\vec{w}) = \vec{x}\vec{w}\vec{z}$ .

For example,  $C := \langle s, ish \rangle$  is a 1-context.  $C(elf) = s^{\wedge}elf^{\wedge}ish = selfish$ . Notice that for any 1-context  $C = \langle \vec{x}, \vec{y} \rangle$ ,  $C(\varepsilon) = \vec{x}\vec{y}$ . The substitution test runs as follows: take a sentence like

(2.30) John likes cricket.

and look for the string occurrences that can be substituted for /cricket/ such that the result is once again a sentence. These include /chess/, /vegetables/, /his new home/ and so on. Similarly, we try to substitute for other sequences such as /likes/, /John likes/ and /likes cricket/. The underlying idea is that nonconstituents cannot be substituted for (for example /John likes/) while constituents can. In practice, this test is not without problems, as it often turns out that nonconstituents can be substituted for (as is the case with /John likes/, for which we can substitute /Peter dislikes/). In fact, it sometimes turns out that the alleged nonconstituent passes all tests and we must be prepared to either strengthen our tests or admit that these really *are* constituents (as some claim is the case with /John likes/, see [Steedman, 1990]). In this section we shall look in some detail at the formal underpinnings of the substitution test.

First of all, we have to ask what we actually mean by substitution and second how it can possibly show us something about the grammars for our language. The answer to the first question is in fact not trivial. In the absence of a grammar the substitution we should be performing is simply *string substitution*. The underlying claim of the constituency test is that it shows us when string substitution is actually *constituent substitution*. This is the case if it can be performed without affecting grammaticality. Here I have *defined* constituent substitution to be substitution on the level of the analysis terms: it is the substitution of one subterm by another. The syntactic tests assume that constituent substitution if defined is always string substitution. This is problematic for two reasons. One is that the two need not be identical because the string functions of the grammar may be different from string polynomials (see the end of this section for a definition). The second is that the substitution can give misleading evidence. We start with some examples to show the point.

**Definition 2.12** *Let  $L$  be a language. Write  $\vec{x} \sim_L \vec{y}$  if for all 1-contexts  $C$ :  $C(\vec{x}) \in L \Leftrightarrow C(\vec{y}) \in L$ . The set  $\text{Cat}_L(\vec{x}) := \{C : C(\vec{x}) \in L\}$  is called the **string category of  $\vec{x}$  in  $L$** .*

Obviously,  $\vec{x} \sim_L \vec{y}$  if and only if  $\text{Cat}_L(\vec{x}) = \text{Cat}_L(\vec{y})$ . If string substitution is constituent substitution then the definition above defines exactly the syntactically relevant classes of English. However, mostly this is not a realistic assumption. Let us review how the notion of part can depart from that of a constituent.

**Example 10.** We look at three grammars to form the plural of English. Let  $F_0$  be a list of functions  $f_x$ , where in all grammars below  $f_x$  will be evaluated to the string  $x$ . To keep it simple, let  $F_0 = R \cup I$ , where  $F_R = \{f_{\text{cat}}, f_{\text{dog}}\}$ ,  $F_I = \{f_{\text{sheep}}, f_{\text{mouse}}, f_{\text{ox}}\}$ . Thus  $F_R$  contains the regular nouns,  $F_I$  the irregular nouns. Thus, with  $\mathcal{J}_i$  the interpretation function of grammar  $i$  we have  $\mathcal{J}_i(f_{\text{cat}})() = \text{cat}$ ,  $\mathcal{J}_i(f_{\text{mouse}})() = \text{mouse}$ , and so on. Now, put  $\Omega_0(f_x) := 0$ . We denote by  $R_s$  the set  $\{x : f_x \in F_R\}$ ,  $R_p$  the corresponding plural forms, likewise  $I_s := \{x : f_x \in F_I\}$ ,  $I_p$  the corresponding plural forms.

The first grammar,  $P_1 = \langle \Omega_1, \mathcal{J}_1 \rangle$ , is as follows.  $F_1 := F_0 \cup \{p\}$ ,  $\Omega_1(p) = 1$ ,  $\Omega_1 \upharpoonright F_0 = \Omega_0$ .  $\mathcal{J}_1(p)$  is defined on  $R_s \cup I_s$ , that is all strings that are singular nouns (/cat/, /mouse/, /ox/, but not /oxen/) and its output is the corresponding plural. So we have

$$(2.31) \quad \mathcal{J}_1(p) = \{\langle \text{cat}, \text{cats} \rangle, \langle \text{dog}, \text{dogs} \rangle, \langle \text{sheep}, \text{sheep} \rangle, \\ \langle \text{mouse}, \text{mice} \rangle, \langle \text{ox}, \text{oxen} \rangle\}$$

The second grammar,  $P_2 = \langle \Omega_2, \mathcal{J}_2 \rangle$ , has instead  $F_2 := F_0 \cup \{g, f_{\text{mice}}, f_{\emptyset}, f_s, f_{\text{es}}, f_{\text{en}}\}$ , where  $g$  is a binary function symbol. We put

$$(2.32) \quad \mathcal{J}_2(g)(\vec{x}, \vec{y}) := \begin{cases} \vec{x}\vec{y} & \text{if } \vec{x} \in R_s \text{ and } \vec{y} = \text{s} \\ & \text{or } \vec{x} = \text{sheep and } \vec{y} = \varepsilon \\ & \text{or } \vec{x} = \text{ox and } \vec{y} = \text{en} \\ \text{undefined} & \text{else.} \end{cases}$$

In short,  $\mathcal{J}_2(g)(\vec{x}, \vec{y})$  is defined only if  $\vec{x}$  is a noun root and  $\vec{y}$  a proper plural suffix for  $\vec{x}$ . Since the plural of /mouse/ is not obtained by affixation, it has been added to the lexicon. A variation of this grammar would be to set  $\mathcal{J}_2(g)(\text{mouse}, \varepsilon) := \text{mice}$ . Thus, the plural is formed by a zero affix to a different stem.

The third grammar,  $P_3 = \langle \Omega_3, \mathcal{J}_3 \rangle$  is a mixture between the two.  $F_3 := F_0 \cup \{p, g, f_s, f_{\text{es}}\}$ . For regular nouns it uses  $g$ , for irregular nouns it uses  $f$ .

$$(2.33) \quad \mathcal{J}_3(g)(\vec{x}, \vec{y}) = \begin{cases} \mathcal{J}_2(g)(\vec{x}, \vec{y}) & \text{if } \vec{x} \in R_s \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$(2.34) \quad \mathcal{J}_3(p)(\vec{x}) = \begin{cases} \mathcal{J}_1(p)(\vec{x}) & \text{if } \vec{x} \in I_s \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$(2.35)$$

First of all notice that we can distinguish between these grammars in terms of the generated language. It turns out that  $P_1$  generates all and only the singular and plural noun forms.  $P_2$  in addition contains the plural morphs (like /s/, /es/, /en/, and  $\varepsilon$ ).  $P_3$  contains only the regular plural morphs and not  $\varepsilon$ , for example (though that depends on the exact labour sharing between  $f$  and  $g$ ).  $P_1$  realises a model called *item and process*, while  $P_2$  realises a model called *item and arrangement* (see [Matthews, 1978] for a discussion of these models).

Next we need to look at how constituent substitution works in these examples. Here is an example: in  $P_2$ , the string /cats/ is the value of the term  $gf_{\text{cat}}f_s$ . Replace  $f_{\text{cat}}$  by  $f_{\text{dog}}$  and you get the term  $gf_{\text{dog}}f_s$ , which unfolds to /dogs/. Replace it by  $f_{\text{mouse}}$  and you get  $gf_{\text{mouse}}f_s$ , which is undefined. Similarly, replace  $f_s$  by  $f_{\text{en}}$  and you get  $gf_{\text{cat}}f_{\text{en}}$ , which also is undefined.

In  $P_2$ , the plural morph is a constituent, so it should be substitutable. Likewise, the root noun is a constituent, so we should be able to substitute for it. Sometimes we can successfully perform such a substitution, as certain nouns accept two plural endings: we have /formulas/ next to /formulae/. Most of the time the substitution will fail, though. In  $P_1$  on the other hand the substitution of the plural morph is illicit for a different reason: it is not a constituent. The form /cats/ is the value of  $pf_{\text{cat}}$ , so the only constituent substitution we can perform is to replace  $f_{\text{cat}}$  by  $f_{\text{mouse}}$ , and in this case the result is /mice/.

In  $P_3$  string substitution of the plural morph by something else is sometimes licit sometimes not. Let us look now at the substitution of the root noun by another root noun. In  $P_2$  we may exchange /house/ for /cat/ but not /mouse/. This is because  $\mathcal{J}_2(g)(\text{house}, s) = \text{houses}$ , which is the result of substituting the substring /cat/ of /cats/ by /house/, but  $\mathcal{J}_2(g)(\text{mouse}, s)$  is undefined, while applying the string substitution gives /mouses/. Trying the same in  $P_1$  we find that the string substitution facts are similar; however,  $\mathcal{J}_2(f)(\text{mouse})$  is defined, and it gives /mice/. Thus, the difference between  $P_1$  and  $P_2$  is that the substitution of the subconstituent /mouse/ for /cat/ in the derivation is licit in  $P_1$ , but illicit in  $P_2$ . In  $P_1$ , the result of this substitution is different from string substitution, though.  $\otimes$

The grammar  $P_2$  actually uses straight concatenation and the string categories of English actually do tell us about the necessary distinctions we need to make in the paradigms. (Note though that the grammars here do not explicitly mention paradigms. There is no need to do so. The classes are just defined indirectly via the partiality.)

**Example 11.** The next example is a variation of the previous theme. The first grammar,  $Q_1$ , has constants for the names /John/, /Alex/, and /Pete/ and for the verb forms, /sings/ and /sing/, /runs/ and /run/. It has two binary functions  $c$ ,  $g$ . Define an **NP** to be a sequence of the form  $x_1\_and\_x_2\_and\_x_3 \dots$ . It is *singular* if it does not contain /and/ and plural otherwise.

$$(2.36) \quad \mathcal{J}(c)(\vec{x}, \vec{y}) := \begin{cases} \vec{x} \_and\_ \vec{y} & \text{if } \vec{x} \text{ and } \vec{y} \text{ are NPs} \\ \text{undefined} & \text{else} \end{cases}$$

$g$  combines NPs with verb forms. The chosen verb form must agree in number with the sequence. This is done as follows.

$$(2.37) \quad \mathcal{J}(g)(x, y) := \begin{cases} x \_and\_ y & \begin{array}{l} \text{if either } x \text{ is a singular NP} \\ \text{and } y \text{ is a singular verb form} \\ \text{or } x \text{ is a plural NP} \\ \text{and } y \text{ is a plural verb form} \end{array} \\ \text{undefined} & \text{else} \end{cases}$$

This grammar generates /John sings/ (it is the value of  $g.f_{\text{John}}.f_{\text{sings}}$ ) and /John and Mary and Alex sing/ (the value of  $g.c.c.f_{\text{John}}.f_{\text{Mary}}.f_{\text{Alex}}.f_{\text{sing}}$ ) but not /Mary and Alex sings/. For the second grammar,  $Q_2$ , we assume we have only verb roots (form identical with the singulars of the previous grammar) and change the interpretation of  $g$  as follows:

$$(2.38) \quad \mathcal{K}(g)(\vec{x}, \vec{y}) := \begin{cases} \vec{x} \_and\_ \vec{y} & \text{if } \vec{x} \text{ is a plural NP and } \vec{y} \text{ is a verb root} \\ \vec{x} \_and\_ \vec{y} \_s & \text{if } \vec{x} \text{ is a singular NP and } \vec{y} \text{ is a verb root} \\ \text{undefined} & \text{else} \end{cases}$$

In  $Q_1$ , we can string substitute /John and Mary/ for /John/ only if the verb form is already plural, but not, for example, in /John sings/, for we would get /John and Mary sings/, which the grammar does not generate. We can also not constituent substitute, for the result is the same. In  $Q_2$ , the constituent substitution gives us different results. Namely, constituent substitution of /John and Mary/ for /John/ in /John sings/ yields /John and Mary sing!/ This is because the sentence is the value (under  $\mathcal{K}$ ) of  $g.f_{\text{John}}.f_{\text{sing}}$ , and we replace  $f_{\text{John}}$  by  $c.f_{\text{John}}.f_{\text{Mary}}$ . This yields the term  $g.c.f_{\text{John}}.f_{\text{Mary}}.f_{\text{sing}}$ , which unfolds to /John and Mary sing/.



The previous examples established two things: first, it may be the case that certain affixes are introduced by the derivation. In this case, the string substitution has nothing to do with constituent substitution, since there is no constituent to begin with. Second, there is a difference between string substitution and constituent substitution. It is the latter notion that is dependent on the grammar. It is defined as follows.

We have seen in the previous section how to evaluate constant terms. Now we shall introduce variables over constituents. Thus, we shall allow to write  $fx$  and  $gxy$  but also  $gfxx$ , where  $f$  is unary and  $g$  binary, and  $x$  and  $y$  are variables over terms. For terms containing such variables the interpretation must be a function from values of 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  $\bar{s} := \langle s_0, s_1, \dots \rangle$  codes the assignment of strings to the variables that maps  $x_i$  to the string  $s_i$ . Now put

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

Again, if  $G$  is clear from the context,  $\iota_G$  will be simplified to  $\iota$ . Notice that if the string functions are partial some of the  $\iota_G(t)$  may also be partial functions. In the sequel I shall not use  $x_0$  and  $x_1$ , but the usual  $x, y$  instead. ( $\iota$  has been defined in Section 2.1 for constant terms slightly differently. On constant terms the valuation is irrelevant.)

**Example 12.** We continue Example 9. The grammar  $C_1$  has only unary functions, so the terms we can create have at most one variable. Examples are  $f_1x_0$ ,  $f_0f_1x_1$ , and so on. These describe functions from assignments to strings. The first defines a function from  $\bar{s}$  to  $A^*$ :  $\bar{s} \mapsto s_0\hat{\ }a$ . The second is  $\bar{s} \mapsto s_1\hat{\ }b$ . I shall simplify this by eliminating reference to the entire valuation and replacing  $s_0$  and  $s_1$  by metavariables. This way we get the somewhat simpler expression  $\vec{x} \mapsto \vec{x}\hat{\ }a$ ,  $\vec{x} \mapsto \vec{x}\hat{\ }b$ . It is possible to describe the totality of definable functions. They all have the form  $\vec{x} \mapsto \vec{x}\hat{\ }y$  for some  $y \in A^*$  (which may be empty, since we generally also have the term  $x$ , which denotes the identity function on  $A^*$ ).



$C_2$  has many more functions. In fact, the terms that we can define in  $C_2$  are all the definable string polynomials using constants from  $A$ .  $\odot$

It is the simplifications of the preceding example that I shall adopt throughout. If we have a term  $t(x, y, z)$  then applying it to some values  $\vec{x}$ ,  $\vec{y}$  and  $\vec{z}$  for  $x$ ,  $y$  and  $z$ , respectively, is denoted by  $t(x, y, z)[\vec{x}, \vec{y}, \vec{z}]$ , or—if we want to make explicit which value replaces which variable—,  $t(x, y, z)[\vec{x}/x, \vec{y}/y, \vec{z}/z]$ . The latter notation is more practical when we suppress the variables in the term itself by writing  $t[\vec{x}/x, \vec{y}/y, \vec{z}/z]$ . Now let  $f : (A^*)^n \rightarrow A^*$ . Say that it is a **term function** of  $G$  if there is a term  $t(x_0, x_1, \dots, x_{n-1})$  such that

$$(2.39) \quad f(\vec{x}_0, \dots, \vec{x}_{n-1}) = \iota_G(t)[\vec{x}_0/x_0, \dots, \vec{x}_{n-1}/x_{n-1}]$$

A **polynomial** (over  $A$ ) is a term in the signature expanded by  $f_a$  (with value  $a$ ) for every  $a \in A$ .  $f$  is a **polynomial function** of  $G$  if there is a polynomial  $p(x_0, x_1, \dots, x_{n-1})$  such that

$$(2.40) \quad f(\vec{x}_0, \dots, \vec{x}_{n-1}) = \iota_G(p)[\vec{x}_0/x_0, \dots, \vec{x}_{n-1}/x_{n-1}]$$

A particular sort of polynomial is the **string polynomial**. Let  $A$  be an alphabet. Then the string polynomials over  $A$  are the polynomials defined over the signature  $\Omega : \cdot \mapsto 2, \varepsilon \mapsto 0$  in the algebra  $\langle A^*, \varepsilon, \wedge \rangle$ . The interpretation is fixed:  $\cdot$  is interpreted by concatenation,  $\varepsilon$  by the empty string and  $a$  by constant yielding the letter  $a$  itself. (Bracketing is therefore eliminable since string concatenation is associative.) For example,  $p(x_0, x_1) := x_1 \cdot a \cdot x_1 \cdot x_0 \cdot b$  is a polynomial. It is interpreted by the following function over  $A^*$ :

$$(2.41) \quad p^{A^*}(\vec{x}, \vec{y}) := \iota_G(t)[\vec{x}/x_0, \vec{y}/x_1] := \vec{y} \wedge a \wedge \vec{y} \wedge \vec{x} \wedge b$$

Typically, we do not even write the dot, so that  $x_0 \cdot x_1$  reduces to  $x_0x_1$ .

I close this chapter with an observation concerning the method of substitution, using Definition 2.12. This test is supposed to reveal something about the structure of the language provided that the grammar for it is some constituent grammar: parts are assumed to be substrings. (If the grammar is not of that form, another form of test is needed.) There are two ways to understand this test, ultimately deriving from two different definitions of language; one is to start with a language as the set of sentences and try to define the constituents smaller than sentences via substitution classes. Another, less ambitious method, starts with a language in the wide sense and tries to find out the constituent *occurrences* in a given string. We

shall look here at the first of these interpretations; the other interpretation shall be looked at in more detail later.

Let  $L \subseteq A^*$  be a language and  $\vec{x}$  a string. Evidently, there are two cases. Either  $\vec{x}$  is not a substring of any string in  $L$ , and so  $\text{Cat}_L(\vec{x}) = \emptyset$ , or it is and then  $\text{Cat}_L(\vec{x}) \neq \emptyset$ . Apart from that there is nothing of substance one can say about the distribution of categories. There is no theoretical instrument to tell us from the substitution possibilities which are the constituents. This is reflected also in some grammars. In the Lambek Calculus all substrings of a string of the language are given a category.

There is a little bit that we can say about the relationship between the number of categories and  $L$  itself. For it turns out that if the set of string categories is finite the language is regular.

**Theorem 2.13 (Myhill, Nerode)** *A language has finitely many string categories if and only if it is regular.*

**Proof.** Suppose that  $L$  has finitely many categories. Intersect the categories with the set  $\{\langle \varepsilon, \vec{x} \rangle : \vec{x} \in A^*\}$ . This yields a finite set of occurrences of prefixes. By the Myhill-Nerode Theorem, the language is regular. Now assume that the language is regular, and accepted by a finite automaton  $\mathfrak{A}$ . Let  $I_i$  be the language of all strings that lead from the initial state to state  $i$ ; and let  $A_j$  be the language of all strings that lead from  $j$  to some accepting state. Then the categories coincide with the sets of pairs  $I_i \times A_j$  for all states  $i$  and  $j$  such that  $j$  can be reached from  $i$ .  $\square$

**Exercise 7.** Describe all unary term functions of  $C_2$ , that is, all actions of  $C_2$ -terms in one variable.

**Exercise 8.** Verify that the language of ua-terms is defined by the following grammar:

$$\begin{aligned}
 & \mathcal{J}(n_0)() := \mathbf{0} \\
 & \quad \dots\dots \\
 & \mathcal{J}(n_9)() := \mathbf{9} \\
 & \mathcal{J}(c_0)(\vec{x}) := \vec{x} \mathbf{0} \\
 (2.42) \quad & \quad \dots\dots \\
 & \mathcal{J}(c_9)(\vec{x}) := \vec{x} \mathbf{9} \\
 & \mathcal{J}(a_0)(\vec{x}) := \vec{x} \mathbf{+} \mathbf{0} \\
 & \quad \dots\dots \\
 & \mathcal{J}(a_9)(\vec{x}) := \vec{x} \mathbf{+} \mathbf{9}
 \end{aligned}$$

**Exercise 9.** (Continuing the previous exercise.) In the grammar of the previous exercise  $/10+1/$  is a part of  $/10+12/$ . Simply choose the analysis  $n_1c_0a_1c_2$ . However,  $/12/$  is not a part of  $/10+12/$  although intuitively it should be. Begin by specifying when a given string is a substring of another. Then write a grammar where only those substring occurrences are parts that should be.

**Exercise 10.** The language of ua-terms is regular. Nevertheless, show that there is no regular grammar that generates exactly this language in the sense that the union of all expressions that belong to the grammar is  $L$ . *Hint.* Regular grammars allow to add only one symbol at a time.

## 2.3 Grammars and String Categories

In the previous section we looked at string categories defined by replacing substrings by other substrings. In this section we look at a similar but different definition where replacement is done only of constituent occurrences. This definition presupposes a grammar.

**Definition 2.14** Let  $G$  be a grammar  $\vec{x}$  and  $\vec{y} \in L(G)$ . We write  $\vec{x} \sim_G \vec{y}$  if for every term  $t(x_0)$ ,  $\iota_G(t)(\vec{x})$  is defined if and only if  $\iota_G(t)(\vec{y})$  is defined. We write  $[\vec{x}]_G := \{\vec{y} : \vec{x} \sim_G \vec{y}\}$ . These sets are called the **syntactic categories** of  $G$ .

We have restricted the definition to strings in  $L(G)$ . Thus, categories are defined only on the strings of the language. Strings outside the language have no category. An alternative formulation is this:  $\vec{x}$  and  $\vec{y}$  have the same category if for every pair of terms  $s_0$  and  $s_1$  that unfold to  $\vec{x}$  and  $\vec{y}$  respectively,  $t(s_0)$  is orthographically definite if and only if  $t(s_1)$  is. (It is easy to see that if this holds for one pair of terms  $s_0$  and  $s_1$  then it holds for all. See also Definition 2.31.)

Notice that the set of strings on which *no* function is defined also is a syntactic category. For example, in Example 1 this category is empty, in Example 6 it contains all equations.

There need not be finitely many equivalence classes as the following example shows.

**Example 13.** Let  $A := \{a\}$ .  $G = \langle \Omega, \mathcal{J} \rangle$  is defined by  $\Omega(e) = 0$ ,  $\Omega(f) = \Omega(g) = 1$  and

$$(2.43) \quad \begin{aligned} \mathcal{J}(e)() &:= \varepsilon \\ \mathcal{J}(f)(a^n) &:= \begin{cases} a^{n-1} & \text{if } n > 0 \\ \text{undefined} & \text{else} \end{cases} \\ \mathcal{J}(g)(a^n) &:= a^{2n} \end{aligned}$$

$G$  generates  $a^*$  in a somewhat unconventional way. In this case we have that if  $m > n$ :  $\mathcal{J}(f)^n(a^m) = a^{m-n}$  and  $\mathcal{J}(f)^m(a^m) = \varepsilon$ . However, for  $n > m$ ,  $\mathcal{J}(f)^n(a^m)$  is undefined. Thus,  $a^m \sim_G a^n$  if and only if  $m = n$ , and so there are infinitely many equivalence classes.

Now, the grammar  $H = \langle \Omega', \mathcal{J} \rangle$  with  $F' := \{e, h\}$  where  $\mathcal{J}(e)() := \varepsilon$ , and  $\mathcal{J}(h)(\vec{x}) := \vec{x}a$  has exactly one class of strings. It is checked that  $a^m \sim_H a^n$  for all  $m, n \in \mathbb{N}$ . ⊛

It is linguistic practice not to leave the categories implicit (in the form of domain restrictions) but to make them part of the representation. If we so wish this can be implemented as follows. Let  $C$  be a set. A **c-string** is a pair  $s = \langle \vec{x}, c \rangle$

where  $\vec{x} \in A^*$  and  $c \in C$ . Given  $s$ , we put

$$(2.44) \quad \varepsilon(s) := \vec{x}, \quad \kappa(s) := c$$

For a set  $S$  of c-strings write  $\varepsilon[S] := \{\varepsilon(s) : s \in S\}$ ,  $\kappa[S] := \{\kappa(s) : s \in S\}$ . A **c-string language** is a subset of  $A^* \times C$ . A **c-string grammar** is a pair  $\langle \Omega, \mathcal{C} \rangle$  where  $\Omega$  is a signature (with domain  $F$ ) and  $\mathcal{C}$  an interpretation function such that for all  $f \in F$   $\mathcal{C}(f) : (A^* \times C)^{\Omega(f)} \hookrightarrow (A^* \times C)$ . We define  $\iota_G(t)$  for an  $\Omega$ -term  $t$  by

$$(2.45) \quad \iota_G(f s_0 \cdots s_{\Omega(f)-1}) := \mathcal{C}(f)(\iota_G(s_0), \cdots, \iota_G(s_{\Omega(f)-1}))$$

We write  $t^\varepsilon$  in place of  $\varepsilon(\iota_G(t))$  and  $t^\kappa$  in place of  $\kappa(\iota_G(t))$ . Thus we have

$$(2.46) \quad \iota_G(t) = \langle t^\varepsilon, t^\kappa \rangle$$

We also use the notation  $f^\varepsilon$  for the function  $\varepsilon \circ \mathcal{C}(f)$  and  $f^\kappa$  for  $\kappa \circ \mathcal{C}(f)$ . A more detailed discussion can be found in Chapter 3. The categories will be most useful when the string operations of the grammar are independent. We shall deal with grammars acting on several components in Chapter 3.

**Example 14.** The shift to categories is not as innocent as it first appears, for we lose certain properties. Here is an example. The relation “is part of” is no longer transitive. Let  $F := \{f_0, f_1, g\}$ ,  $\Omega(f_0) := \Omega(f_1) := 0$  and  $\Omega(g) := 1$ .  $C := \{\alpha, \beta\}$  and  $A := \{a\}$ .

$$(2.47) \quad \begin{aligned} \mathcal{J}(f_0)(\langle \vec{x}, c \rangle) &:= \langle a, \alpha \rangle \\ \mathcal{J}(f_1)(\langle \vec{x}, c \rangle) &:= \langle aa, \alpha \rangle \\ \mathcal{J}(g)(\langle \vec{x}, c \rangle) &:= \begin{cases} \langle \vec{x} \hat{\ } a, \beta \rangle & \text{if } c = \alpha \\ \text{undefined} & \text{else} \end{cases} \end{aligned}$$

This grammar generates the language  $\{\langle a, \alpha \rangle, \langle aa, \beta \rangle, \langle aa, \alpha \rangle, \langle aaa, \beta \rangle\}$ . It turns out that  $/a/$  is a part of  $/aa/$ , and  $/aa/$  a part of  $/aaa/$ , but  $/a/$  is not a part of  $/aaa/$ .



As the example shows we can no longer simply say that a string occurs as a substring; it occurs in a c-string as a c-string and so the category that it has may also be fixed. Just like we say that `/I see John fly./` contains `/fly/` as a verb and not as a noun.

An important class of c-string grammars are the *context free grammars* (CFGs). These are not the same as ordinary CFGs. It is therefore necessary to repeat the standard definition of CFGs first and then return to the c-string grammars. Recall that a context free grammar is standardly taken to be a quadruple  $G = \langle A, N, S, R \rangle$ , where  $A \cap N = \emptyset$  are disjoint sets,  $S \in N$ , and  $R$  a set of replacement rules. They have the form  $X \rightarrow \vec{y}$ , where  $X \in N$  and  $\vec{y} \in (A \cup N)^*$  is a sequences over  $A \cup N$ . The rules define a replacement relation in the following way.

**Definition 2.15** *If  $\rho = \vec{x} \rightarrow \vec{y}$  is a rule, then we say that  $\vec{u}\vec{y}\vec{w}$  is **1-step derivable** from  $\vec{u}\vec{x}\vec{v}$ , in symbols  $\vec{u}\vec{x}\vec{v} \Rightarrow_{\rho} \vec{u}\vec{y}\vec{v}$ . For a set  $R$  of rules we write  $\vec{u}\vec{x}\vec{v} \Rightarrow_{\rho} \vec{u}\vec{y}\vec{v}$  and say that  $\vec{u}\vec{y}\vec{v}$  is **1-step derivable** from  $\vec{u}\vec{x}\vec{v}$  if there is a rule  $\rho \in R$  such that  $\vec{u}\vec{x}\vec{v} \Rightarrow_{\rho} \vec{u}\vec{y}\vec{v}$ . Furthermore, we say that  $\vec{w}$  is  **$n$ -step derivable** in  $R$  from  $\vec{v}$  and write  $\vec{v}\vec{x}\vec{v} \Rightarrow_R^n \vec{w}$  if either  $n = 0$  and  $\vec{w} = \vec{v}$  or  $n > 0$  and there is a  $u$  such that  $\vec{u}$  is  $n - 1$ -step derivable from  $\vec{v}$  and  $\vec{w}$  is 1-step derivable from  $\vec{u}$ .*

Notice that  $\vec{v} \Rightarrow_R^1 \vec{w}$  and  $\vec{v} \Rightarrow_R \vec{w}$  are synonymous, and that  $\vec{v} \Rightarrow_{\{\rho\}} \vec{w}$  and  $\vec{v} \Rightarrow_{\rho} \vec{w}$  are also synonymous;  $R$  or  $\rho$  will be dropped when the context makes clear which rule(s) are being used. Notice furthermore that it may happen that a rule can be applied to a given string in several ways. The rule  $A \rightarrow aa$  can be applied to the string  $AcAb$  to yield either  $aacAb$  or  $Acaab$ . Therefore, if we want to know what the next result will when applying the rule we need to identify the occurrence of the left-hand side that is being replaced. When can do this as follows:  $\underline{A}cAb \Rightarrow aacAb$ .  $Ac\underline{A}b \Rightarrow Acaab$ . If the occurrence is underlined then the rule *must* be applied to the underlined occurrence. Hence we do not have  $\underline{A}cAb \Rightarrow Acaab$ . Conversely, suppose we have a marked string; then the result is still not unique unless we know which rule is being applied. This follows from the fact that several rules may replace a given string. For example, if we also have the rule  $A \rightarrow cd$  then from  $\underline{A}cAb$  we may proceed to  $cdcAb$  in addition to  $aacAb$ . However, if also the resulting string is given the rule that has been applied can be inferred. Thus, in order to show that a given string  $\vec{w}$  is  $n$ -step derivable from a string  $\vec{v}$  we need to produce a sequence  $\langle \vec{v}_i : i < n \rangle$  of length  $n$  of underlined strings such that  $\vec{v}_i \Rightarrow \vec{v}_{i+1}$  for  $i < n - 1$  and  $\vec{v}_{n-1} \Rightarrow \vec{w}$ . Such a sequence is called a **derivation**. Notice that the sequence contains marked strings not just strings, though we shall often not show the mark. The derived string is by definition not marked, though it is often added at the end of the derivation sequence so that one can infer the choice of rules in each step.

Given a nonterminal  $A$  and a string  $\vec{x}$  we write  $A \vdash_G \vec{x}$  if and say that  $G$  **derives**

$\vec{x}$  from  $A$  if there is an  $n$  such that  $S \Rightarrow_R^n \vec{x}$ , where  $S$  is the start symbol and  $R$  the rule set of  $G$ . The language of  $G$  is defined by

$$(2.48) \quad L(G) := \{\vec{x} \in A^* : S \vdash_G \vec{x}\}$$

A language is **context free** if there is a context free grammar  $G$  such that  $L = L(G)$ . Also, write  $[A]_G := \{\vec{x} : A \vdash_G \vec{x}\}$ . Then  $L(G) = [S]_G$ . This notion of grammar is top down and nondeterministic. It generates the strings from a single string (consisting in the single letter  $S$ ).

**Example 15.** Let  $G$  be defined as follows.

$$(2.49) \quad G := \langle \{a, \dots, z, \_ \}, \{ \langle S \rangle, \langle NP \rangle, \langle VP \rangle, \langle N \rangle, \langle D \rangle, \langle VI \rangle, \langle VT \rangle \}, \langle S \rangle, R \rangle$$

The alphabet consists of all lower case letters plus the space.

$$(2.50) \quad R = \{ \begin{array}{l} \langle S \rangle \rightarrow \langle NP \rangle \langle VP \rangle \\ \langle NP \rangle \rightarrow \langle D \rangle \langle N \rangle \\ \langle D \rangle \rightarrow \text{the\_} \mid \text{a\_} \\ \langle N \rangle \rightarrow \text{cat\_} \mid \text{dog\_} \mid \text{mouse\_} \\ \langle VP \rangle \rightarrow \langle VI \rangle \mid \langle VT \rangle \langle NP \rangle \\ \langle VI \rangle \rightarrow \text{runs\_} \mid \text{sleeps\_} \\ \langle VT \rangle \rightarrow \text{sees\_} \mid \text{chases\_} \end{array} \}$$

This grammar generates among other the following strings:

$$(2.51) \quad \begin{array}{l} \langle S \rangle \\ \langle NP \rangle \langle VP \rangle \\ \langle D \rangle \text{dog\_} \langle VI \rangle \\ \text{a\_} \text{dog\_} \text{chases\_} \text{the\_} \text{cat} \end{array}$$

Only the last of the four strings is meaningful. A derivation of the third string is as follows.

$$(2.52) \quad \langle \underline{\langle S \rangle}, \underline{\langle NP \rangle} \langle VP \rangle, \langle D \rangle \langle N \rangle \underline{\langle VP \rangle}, \langle D \rangle \underline{\langle N \rangle} \langle VI \rangle \langle D \rangle \text{dog\_} \langle VI \rangle \rangle$$



In our present system we need a definition of a CFG that is bottom up. This is because the nonterminals are meaningless symbols and must be eliminated. To get at a bottom up version we turn the rules around. Rather than assuming a rule, say,  $\rho = A \rightarrow BC$ , we define a string function  $f_\rho$  of arity 2 such that  $f_\rho$  is interpreted as concatenation, that is,  $\mathcal{J}(f_\rho)(\vec{x}, \vec{y}) = \vec{x}\vec{y}$ . However, this function is only defined if  $\vec{x}$  is a  $B$ -string, that is, if we can derive  $\vec{x}$  from  $B$  in the grammar and if  $\vec{y}$  is a  $C$  string. In this way we guarantee that  $\vec{x}\vec{y}$  is an  $A$ -string. In general, for each rule  $\rho$  we assume a function symbol  $f_\rho$  and an interpretation  $\mathcal{J}(f_\rho)$ . A rule is of the form  $A \rightarrow \vec{x}$  for some  $\vec{x} \in (A \cup N)^*$ .

This means that there is  $n$  and  $\vec{x}_i \in A^*$ ,  $i < n + 1$ , and  $B_i \in N$ ,  $i < n$ , such that

$$(2.53) \quad \rho = A \rightarrow \vec{x}_0 \vec{B}_0 \vec{x}_1 B_1 \cdots B_{n-1} \vec{x}_n$$

Then  $\Omega(f_\rho) := n$ , and its interpretation is

$$(2.54) \quad \mathcal{J}(f_\rho)(\vec{y}_0, \dots, \vec{y}_{n-1}) := \begin{cases} \vec{x}_0 \vec{y}_0 \vec{x}_1 \vec{y}_1 \cdots \vec{y}_{n-1} \vec{x}_n & \text{if for all } i < n: \\ & \vec{y}_i \text{ is a } B_i\text{-string} \\ \text{undefined} & \text{else} \end{cases}$$

We do this for all  $\rho$  which do not have the form  $A \rightarrow B$ . It is an easy matter to transform  $G$  into a grammar that has no such rules, but that transformation is actually unnecessary. This defines the grammar  $G^\diamond$ .

**Example 16.** I transform the grammar from Example 15. Let us note that the constituents generate only finitely many strings, so we can list them all.

$$(2.55) \quad \begin{aligned} [ <D> ]_G &= \{ /a\_, /the\_\} \\ [ <N> ]_G &= \{ /cat\_, /dog\_, /mouse\_\} \\ [ <VI> ]_G &= \{ /runs\_, /sleeps\_\} \\ [ <VT> ]_G &= \{ /sees\_, /chases\_\} \\ [ <VP> ]_G &= (runs \mid sleeps) \mid (sees\_\ \mid chases\_\)(the\_\ \mid a\_) \\ &\quad (cat\_\ \mid dog\_\ \mid mouse\_) \end{aligned}$$

Before transformation we need to consider the rule  $<VP> \rightarrow <VI>$ . This is a unary rule. We eliminate it and add instead the rule

$$(2.56) \quad <S> \rightarrow <NP><VI>$$



Now we begin the transformation. The grammar  $G^\blacklozenge$  is based on the set  $\{f_1, f_2, \dots, f_{11}\}$  with  $\Omega(f_i) = 0$  for  $i < 9$  and  $\Omega(f_i) = 2$  otherwise. We have

$$(2.57) \quad \begin{aligned} \mathcal{J}(f_0)() &:= a\_ \\ \mathcal{J}(f_1)() &:= \text{the}\_ \\ \mathcal{J}(f_2)() &:= \text{cat}\_ \\ \mathcal{J}(f_3)() &:= \text{dog}\_ \\ \mathcal{J}(f_4)() &:= \text{mouse}\_ \\ \mathcal{J}(f_5)() &:= \text{runs}\_ \\ \mathcal{J}(f_6)() &:= \text{sleeps}\_ \\ \mathcal{J}(f_7)() &:= \text{sees}\_ \\ \mathcal{J}(f_8)() &:= \text{chases}\_ \\ \mathcal{J}(f_9)(\vec{x}, \vec{y}) &:= \begin{cases} \vec{x}\vec{y} & \text{if } \vec{x} \in [ <D> ]_G \text{ and } \vec{y} \in [ <N> ]_G \\ \text{undefined} & \text{otherwise} \end{cases} \\ \mathcal{J}(f_{10})(\vec{x}, \vec{y}) &:= \begin{cases} \vec{x}\vec{y} & \text{if } \vec{x} \in [ <VT> ]_G \text{ and } \vec{y} \in [ <NP> ]_G \\ \text{undefined} & \text{otherwise} \end{cases} \\ \mathcal{J}(f_{11})(\vec{x}, \vec{y}) &:= \begin{cases} \vec{x}\vec{y} & \text{if } \vec{x} \in [ <NP> ]_G \text{ and } \vec{y} \in [ <VI> ]_G \\ \text{undefined} & \text{otherwise} \end{cases} \end{aligned}$$

The reader is asked to check that these modes correspond exactly to the rules of the grammar (in its slight modification). The string  $/a\_cat\_sees\_the\_dog\_/_$  is derived by the term  $f_7 f_9 f_0 f_2 f_9 f_1 f_3$ , as can be checked.  $\odot$

$G^\blacklozenge$  is a grammar in the sense of Section 2.1. The grammar  $G^\blacklozenge$  generates the language of  $G$  in the wide sense, as the following theorem documents.

**Proposition 2.16** *Let  $G$  be a context free grammar. Then  $\vec{x} \in L(G^\blacklozenge)$  if and only if there is  $X \in N$  such that  $X \vdash_G \vec{x}$ .*

The proof is a relatively easy induction on the length of derivations. I shall relegate this to the exercises.

**Example 17.** The elimination of unary rules is not as innocent as it first appears. In natural languages there are plenty of examples of zero-derivation. One example is the conversion of adjectives to nouns in Hungarian. Typically, adjectives do not inflect. However, in the absence of a noun they can inflect just as nouns and hence

should be regarded as such. Thus, the form /fehéret/ (accusative from /fehér/) must be translated as ‘a white one’. Critically, also the nominative form /fehér/ can be so regarded and hence be translated as either ‘white’ or ‘a white one’. Given a bottom up grammar these two are not confused. However, as long as we do not treat meaning in addition there is no harm in this. This theme will be picked up in Section 3.4.  $\odot$

Notice that there is no way to generate *only* the language  $L(G)$ , that is, all and only the  $S$ -strings for the start symbol  $S$ . When we do a top down generation we can simply choose to start with the start symbol and all the strings we generate are sentences. However, in the bottom up process we cannot restrict ourselves to generating just the sentences. We must generate all intermediate strings. On the other hand there is no need to generate strings with extraneous symbols. In the c-string grammar we can make up for this defect as follows. For a CFG in the standard sense let

$$(2.58) \quad L^c(G) := \{\langle \vec{x}, X \rangle : X \in N, X \vdash_G \vec{x}\}$$

So,  $L^c(G)$  contains strings together with their categorial information; it does not however single out a particular category. We can derive  $L(G)$  from  $L^c(G)$  by picking all  $\vec{x}$  for which  $\langle \vec{x}, S \rangle \in L^c(G)$ . This is a different notion of language than the generated language in the wide sense. For in the latter we do not know what the categories of the strings are; we just know that they have *some* category. On the other hand, for a language in the wide sense there is no need to construct the categories from the input data (as languages mostly do not always mark their expressions for category). The arity of  $f_\rho$  equals the number of nonterminals on the right hand side of the rule.

The string based version presented above is not an exact equivalent of the grammar  $G$ . In the exercises we shall show that these grammars may have quite different derivations. To get a more exact correspondence we turn to c-strings. In the case at hand we choose  $C := N$ . Thus c-strings are pairs  $\langle \vec{x}, A \rangle$  where  $\vec{x} \in A^*$  and  $A \in N$ . The interpretation of the function symbol  $f_\rho$  is now the partial function

$$(2.59) \quad \begin{aligned} \mathcal{C}(f_\rho)(\langle \vec{y}_0, c_0 \rangle, \langle \vec{y}_1, c_1 \rangle, \dots, \langle \vec{y}_{n-1}, c_{n-1} \rangle) \\ := \langle \vec{x}_0 \vec{y}_0 \vec{x}_1 \vec{y}_1 \cdots \vec{y}_{n-1} \vec{x}_n, f_*^k(c_0, c_1, \dots, c_{n-1}) \rangle \end{aligned}$$

where

$$(2.60) \quad f_*^k(c_0, \dots, c_{n-1}) := \begin{cases} A & \text{if for all } i < n: c_i = B_i \\ \text{undefined} & \text{else} \end{cases}$$

Figure 2.1: A derivation in  $G_Q$ 

$$\begin{array}{l}
\underline{E} \\
\underline{T}=T \\
\underline{B}=T \\
\underline{0}=T \\
0=(T-\underline{T}) \\
0=(T-\underline{B}) \\
0=(\underline{T}-0) \\
0=(\underline{B}-0) \\
0=(\underline{B0}-0) \\
0=(10-0)
\end{array}$$

Then  $L(G)$  is a set of pairs  $\langle \vec{x}, c \rangle$ . We say that  $\vec{x}$  **has category  $c$  in  $G$**  if some  $G$ -term unfolds to  $\langle \vec{x}, c \rangle$ . A given string can have several categories.

**Example 18.** We continue the language of equations (Example 6 on Page 27). The grammar  $G_Q$  consists of the alphabet of terminals


$$(2.61) \quad \text{bt} := \{0, 1, +, -, (, ), =\},$$

The alphabet of nonterminals is  $N := \{E, B, T\}$ , the start symbol  $/E/$  and the set of rules is as follows.

$$\begin{array}{l}
E \rightarrow T=T \\
(2.62) \quad T \rightarrow (T+T) \mid (T-T) \mid B \\
B \rightarrow B0 \mid B1 \mid 0 \mid 1
\end{array}$$

By default, a derivation starts with the letter  $/E/$ . Thus

$$(2.63) \quad C = \langle \text{bt}, N, E, R \rangle$$

Recall that ‘|’ is an abbreviation. It allows to group together rules with the same left hand side. Figure 2.1 shows an example of a derivation in  $G_Q$ . In each step we replace a single occurrence of a nonterminal by a corresponding right hand side of (2.62). 

An  **$X$ -derivation** is a sequence of strings starting with the nonterminal  $X$ , where each nonfirst member is obtained from the previous by replacing a nonterminal symbol in the appropriate way. A **derivation** is an  $X$ -derivation with  $X$  the top symbol. For our purposes, however, the best objects to deal with are not the derivations, but the analysis terms. The analysis term of a derivation is obtained as follows. Assign to each rule  $\rho$  with  $n(\rho)$  nonterminals on the right a function symbol  $f_\rho$  of arity  $n(\rho)$ . This is the signature. Start with the variable  $x_0$ . A step in the derivation consists in the replacement of an occurrence of a variable  $x_i$  by a term of the form  $f_\rho(x_{i_0}, x_{i_1}, \dots, x_{i_{n(\rho)-1}})$  and the  $x_{i_j}$  are not already used. This procedure is best explained with the derivation above.

**Example 19.** Continuing Example 18. We give the following names to the rules.

$$(2.64) \quad \begin{array}{l} a \quad E \rightarrow T=T \\ b \quad T \rightarrow (T+T) \\ c \quad T \rightarrow (T-T) \\ d \quad T \rightarrow B \\ e \quad B \rightarrow B0 \\ f \quad B \rightarrow B1 \\ g \quad B \rightarrow 0 \\ h \quad B \rightarrow 1 \end{array}$$

Thus the symbols are called  $f_a, f_b, f_c$  (binary),  $f_d, f_e, f_f$  (unary), and  $f_g$  and  $f_h$  (zeroary). The derivation is translated to a term as shown in Figure 2.2. The variable that is being replaced is surrounded by a box. The exact recipe is this: if the derivation replaces the  $n$ th nonterminal counting from the left, then it is the  $n$ th variable from the left that is being replaced irrespective of its index.  $\otimes$

Now we shall supply the term symbols with interpretations that match the effect of the rules. Call  $\vec{x}$  an  **$X$ -string** if  $X \vdash_G^* \vec{x}$ . 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 0 and 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  $\{0, 1\}^+$ .

Figure 2.2: Deriving the term

$\underline{\mathbf{E}}$	$\boxed{x_0}$
$\underline{\mathbf{T}}=\mathbf{T}$	$f_a \boxed{x_0} x_1$
$\underline{\mathbf{B}}=\mathbf{T}$	$f_a f_d \boxed{x_0} x_1$
$\underline{\mathbf{0}}=\mathbf{T}$	$f_a f_d f_g \boxed{x_1}$
$\mathbf{0}=(\mathbf{T}-\underline{\mathbf{T}})$	$f_a f_d f_g f_c x_0 \boxed{x_1}$
$\mathbf{0}=(\mathbf{T}-\underline{\mathbf{B}})$	$f_a f_d f_g f_c x_0 f_d \boxed{x_1}$
$\mathbf{0}=(\mathbf{T}-\underline{\mathbf{0}})$	$f_a f_d f_g f_c \boxed{x_0} f_d f_g$
$\mathbf{0}=(\underline{\mathbf{B}}-\mathbf{0})$	$f_a f_d f_g f_c f_d \boxed{x_0} f_d f_g$
$\mathbf{0}=(\underline{\mathbf{B1}}-\mathbf{0})$	$f_a f_d f_g f_c f_d f_e \boxed{x_0} f_d f_g$
$\mathbf{0}=(\mathbf{10}-\mathbf{0})$	$f_a f_d f_g f_c f_d f_e f_h f_d f_g$

For example,  $\rho = \mathbf{B} \rightarrow \mathbf{B1}$  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 /1/. Hence:

$$(2.65) \quad \iota(f_\rho)(\vec{x}) := \begin{cases} \vec{x} \hat{\ } 1 & \text{if } \vec{x} \text{ is a B-string} \\ \text{undefined} & \text{else} \end{cases}$$

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:

$$(2.66) \quad \iota(f_{\rho'})(\vec{x}, \vec{y}) := \begin{cases} (\vec{x} \hat{\ } + \vec{y} \hat{\ }) & \text{if } \vec{x} \text{ and } \vec{y} \text{ are T-strings} \\ \text{undefined} & \text{else} \end{cases}$$

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

We can use Example 19 to show that the transformation of CFGs preserves the strings but not the set of terms. The rule  $d$  has the form  $\mathbf{T} \rightarrow \mathbf{B}$ . It is converted into the string function  $\mathcal{J}(f_d)(\vec{x}) = \vec{x}$ , in other words the identity function. This function is iterable, while the rule is not. Thus the term  $f_d f_d f_g$  evaluates in  $\sigma(G_Q)$  to /0/:

$$(2.67) \quad \iota(f_d f_d f_g) = \mathcal{J}(f_d)(\mathcal{J}(f_d)(\mathcal{J}(f_g)(\mathbf{0}))) = \mathcal{J}(f_d)(\mathcal{J}(f_g)(\mathbf{0})) = \mathcal{J}(f_d)(\mathbf{0}) = \mathbf{0}$$

However, there is no derivation with term  $f_d f_d f_g$ . Try to start with the symbol T,

for example:

$$(2.68) \quad \begin{array}{ll} \text{T} & x_0 \\ \text{B} & f_a x_0 \\ ? & f_a f_a x_0 \end{array}$$

Similarly if we start with /B/. (If you want a derivation beginning with the start symbol, take the term  $f_a f_a f_a f_g f_a f_h$ .) It might be deemed that all we have to do is to exclude unary rules. That this is not so is shown in Exercise 16.

We can characterise in more exact terms the connection between the two kinds of grammars. Here is a characterisation of context free languages in terms of the generating functions. It shows that if the functions are partial functions of a certain kind and such that ranges of functions are subsets of domains (or disjoint) then the generated language is context free (and conversely).

**Definition 2.17** *Let  $G = \langle \Omega, \mathcal{J} \rangle$  be a grammar.  $G$  is called a **concatenation grammar** if for all modes  $f$ ,  $\mathcal{J}(f)$  is the restriction of a polynomial function of the string algebra to some arbitrary set of sequences of strings.*

This definition says the following. In a concatenation grammar a mode  $f$  interpreted as a partial function  $\mathcal{J}(f) : (A^*)^{\Omega(f)} \hookrightarrow A^*$ . While the domain is some arbitrary set  $D \subseteq (A^*)^{\Omega(f)}$ , there must exist a polynomial function  $p$  such that  $\mathcal{J}(f) = p \upharpoonright D$ . Notice namely that the string polynomials are total. These polynomials may be arbitrarily restricted. However, as we shall see, in context free grammars there are tight restrictions on these domains. Say a polynomial  $p(\vec{x})$  is a **linear string polynomial** if it is composed from the variables  $x_i$  and constants such that each  $x_i$  occurs exactly once. If  $p$  is a polynomial, we denote the induced function by  $p^{A^*}$ .  $f : (A^*)^n \rightarrow A^*$  is a **rectangularly restricted linear string polynomial** if there is a linear string polynomial  $p(x_0, \dots, x_{n-1})$  such that  $f \subseteq p^{A^*}(\vec{x})$  and there are subsets  $P_i \subseteq A^*$ ,  $i < n$ , such that  $\text{dom}(f) = \prod_{i < n} P_i$ . Now recall that the grammar  $\sigma(G)$  uses precisely such functions. Thus we have

**Proposition 2.18** *If a language  $L \subseteq A^*$  is context free then it has a grammar  $G$  in which all function symbols are interpreted by rectangularly restricted linear string polynomials.*

For the converse, a little more is needed. Namely, let  $H$  be a grammar such that all  $\mathcal{J}(f)$  are rectangularly restricted linear polynomials. So for each  $f$  there are sets

$Q_i^f$ ,  $i < \Omega(f)$ , such that the domain of  $\mathcal{J}(f)$  is  $\prod_{i < \Omega(f)} Q_i^f$ . Assume moreover that for every  $g$  and  $i < \Omega(g)$ : either  $\text{rng}(\mathcal{J}(f)) \subseteq Q_i^g$  or  $\text{rng}(\mathcal{J}(f)) \cap Q_i^g = \emptyset$ . We call this the **connectivity property** for  $H$ . For each domain  $Q$  we choose a nonterminal  $N_Q$  (notice that  $N_Q = N_P$  if  $P = Q$  as sets). Further, for a function symbol  $f$  such that  $\text{dom}(\mathcal{J}(f)) = \prod_{i < \Omega(f)} Q_i^f$  and  $\text{rng}(\mathcal{J}(f)) \subseteq Q_i^g$  we create a rule

$$(2.69) \quad \rho_f : N_{Q_i^g} \rightarrow \vec{x}_0 N_{Q_0^f} \vec{x}_1 N_{Q_1^f} \vec{x}_2 \cdots \vec{x}_{\Omega(f)-1} N_{Q_{\Omega(f)-1}^f} \vec{x}_{\Omega(f)}$$

where the  $\vec{x}_i$  are chosen such that  $\mathcal{J}(f)$  is the restriction of the polynomial

$$(2.70) \quad p^{A^*}(y_0, \dots, y_{\Omega(f)-1}) := \vec{x}_0 y_0 \vec{x}_1 y_1 \vec{x}_2 \cdots \vec{x}_{\Omega(f)-1} y_{\Omega(f)-1} \vec{x}_{\Omega(f)}$$

This grammar is such that  $\vec{y}$  is an  $N_Q$ -string for some  $Q$  if and only if it is in  $L(H)$ .

**Proposition 2.19** *If  $H$  is a grammar such that all  $\mathcal{J}(f)$  are rectangularly restricted linear string polynomials and  $\mathcal{J}$  has the connectivity property then  $L(H)$  is context free.*

**Example 20.** I give some examples to show that none of the conditions can be dropped. First, the functions must be linear string polynomials. For take  $f(\vec{x}) := \vec{x}\vec{x}$  on the alphabet  $\{a\}$ . This function is induced by the polynomial  $p(x_0) := x_0 x_0$ . It is not linear as the variable  $x_0$  occurs twice on the right. As it happens the function generates the language  $\{a^{2^n} : n \in \mathbb{N}, n > 0\}$  from  $a$ . (Thus, add a constant  $c$  to the signature with interpretation  $a$ .) We can change that use instead the function

$$(2.71) \quad f(\vec{x}, \vec{y}) := \begin{cases} \vec{x}\vec{y} & \text{if } |\vec{x}| = |\vec{y}| \\ \text{undefined} & \text{otherwise.} \end{cases}$$

This (binary) function is the restriction of the polynomial  $p(x_0, x_1) := x_0 x_1$  to the set of all pairs of strings of equal length. Unfortunately, this function is not rectangularly restricted. There are no sets  $H, K$  such that the domain of  $f$  is  $H \times K$ . And the set of strings generable from  $a$  with this function is again the set  $\{a^{2^n} : n \in \mathbb{N}, n > 0\}$ . Finally, consider the following two functions. The first is a modification of  $f$ :

$$(2.72) \quad f(\vec{x}, \vec{y}) := \begin{cases} \vec{x}\vec{y} & \text{if } \vec{x}, \vec{y} \in a^* \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The second is a unary function  $g$  defined by

$$(2.73) \quad g(\vec{x}) := \begin{cases} \vec{x}\mathbf{b} & \text{if } |\vec{x}| = 2^n \text{ for some } n \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Both these functions restrictions of linear polynomial functions to some rectangles. Only the connectivity property is lacking. The generated language is

$$(2.74) \quad \mathbf{a}^+ \cup \{\mathbf{a}^{2^n} : n \in \mathbb{N}, n > 0\}$$

Hence all the conditions are really necessary and independent of each other.  $\otimes$

This gives rise to the following definition.

**Definition 2.20** *A string grammar is called **context free** if it is a concatenation grammar with rectangularly restricted linear string polynomials with the connectivity property.*

Notice that “context free” is applied not only to rule based grammars, but also to c-string grammars and string grammars alike.

I close this section with some remarks concerning the use of categories as discriminatory devices. Suppose two strings are such that in a language they have the same category. Then we will want to say that they should also be of the same category in the analysing grammar. Recall that in a context free language, the formal concept of identity of category was substitutability in all 1-contexts, written  $\vec{x} \sim_L \vec{y}$ .

**Principle 1 (Identity of Indiscernibles)** *Let  $G$  be a context free c-grammar. If  $\vec{x} \sim_L \vec{y}$  and  $\langle \vec{x}, c \rangle \in L(G)$  then also  $\langle \vec{y}, d \rangle \in L(G)$ .*

We shall not spell out the generalisation to other kinds of grammars, though it is straightforward to do.

**Exercise 11.** In Example 14 it was shown that the relation *is a part of* is not transitive. Find an example to show that it is also not antisymmetric. (A relation  $R$  is **antisymmetric** if from  $x R y$  and  $y R x$  follows  $x = y$ .)



**Exercise 12.** A grammar is left regular if the functions are zeroary or unary; and the unary functions all have the form  $f(\vec{x}) := \vec{x} \hat{\ } a$  for some  $a$ . Let  $L$  be a language. Define  $\vec{x}/L := \{\vec{y} : \vec{x} \hat{\ } \vec{y} \in L\}$ . Show that for a regular grammar  $G$  generating  $L$ :  $\vec{x} \sim_G \vec{y}$  if and only if  $\vec{x}/L = \vec{y}/L$ .

**Exercise 13.** Why does the bottom up grammar  $G^\blacklozenge$  not contain any  $f_\rho$  for rules of the form  $\rho = A \rightarrow B$ ?

**Exercise 14.** Let  $G$  be a context free grammar and  $A$  a nonterminal. Let  $H_A := \{\vec{x} : A \vdash_G \vec{x}\}$ . Show that for every  $\vec{x} \in H_A$   $H_A \subseteq [\vec{x}]_G$ . Give an example to show that equality need not hold!

**Exercise 15.** Prove Proposition 2.16.

**Exercise 16.** Context free grammars allow to tune derivations more finely than grammars in the sense of Definition 2.4. Here is an example, due to Ben George. Let  $G$  consist of the rules

$$\begin{aligned} & S \rightarrow L \mid R \\ (2.75) \quad & L \rightarrow La \mid a \\ & R \rightarrow aR \mid a \end{aligned}$$

Construct the corresponding grammar and show that it allows for more analysis terms for the string  $/aaa/$  than does  $G$ .

## 2.4 Indeterminacy and Adjunction

In the previous section we have constructed a “bottom up” version  $G^\blacklozenge$  of a context free grammar  $G$ . (I should stress here, though, that only  $G^\blacklozenge$ , not  $G$ , is a grammar in the sense of this book.) In addition to the differences between these types of grammars that I mentioned earlier there is a major difference between  $G$  and  $G^\blacklozenge$ . It is that by definition  $L(G^\blacklozenge)$  is the set of all strings that are constituents for *some* nonterminals as opposed to just the strings corresponding to the start symbol.

Thus the standard definition of  $L(G)$  for a CFG contains  $L(G^\blacklozenge)$ , but the two need not be identical. The difference is exactly between language in the wide sense and language in the narrow sense. Since I insist that the language of a grammar must be taken in the wide sense we must ask if there is a kind of grammar that generates the sentences all by themselves so that the two notions actually coincide for this type of grammar. Such grammars do exist. The adjunction grammars are of this kind. Unfortunately, these grammars turn out to be somewhat different from the grammars previously defined in that they are *relational*. Grammars of this form shall be called *indeterminate grammars* (the label *relational grammar* has already been taken). I shall return to indeterminate grammars again in Section 3.7 in connection with interpreted languages.

**Definition 2.21** An *indeterminate grammar over  $A$*  is a pair  $\langle \Omega, \mathcal{J} \rangle$ , where  $\Omega$  is a signature and for every  $f \in F$ ,  $\mathcal{J}(f) \subseteq (A^*)^{\Omega(f)+1}$ .  $F$  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**. The **language** generated by  $G$ , in symbols  $L(G)$ , is defined to be the least set  $S$  satisfying for every  $f \in F$  and all  $\vec{x}_i \in A^*$ ,  $i < \Omega(f)$ :

$$(2.76) \quad \text{If for all } i < \Omega(f) : \vec{x}_i \in S \text{ and if } \langle \vec{x}_0, \dots, \vec{x}_{\Omega(f)-1}, \vec{y} \rangle \in \mathcal{J}(f) \text{ then } \vec{y} \in S$$

Thus, the output of a rule is not assumed to be unique. In a grammar of the usual sort the output need not exist, but if it exists, it is unique. In an indeterminate grammar it need not even be unique. Adjunction grammars are such grammars. They are popular since they generate more than context free languages and enjoy nevertheless quite a simple description. I point out that as soon as we move to interpreted languages it will turn out that the indeterminacy will have to be eliminated; see also the discussion in Section 3.7.

**Definition 2.22** 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:

$$(2.77) \quad \gamma(\langle \vec{x}, \vec{y} \rangle) := \vec{u}\vec{x}\vec{v}\vec{y}\vec{w}$$

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

According to the previous definition, the string relation associated with  $\rho$  is

$$(2.78) \quad \text{Adj}(\rho) := \{ \langle \vec{u}\vec{v}\vec{w}, \vec{u}\vec{x}\vec{v}\vec{y}\vec{w} : \langle \vec{x}, \vec{y} \rangle \in \Lambda \}$$

**Definition 2.23** 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.

For a string adjunction grammar we define the following signature: let  $f_{\vec{x}}$  be a symbol of arity 0 for every  $\vec{x} \in S$ ; and let  $g_\rho$  be a symbol of arity 1 for every  $\rho \in R$ . This defines the signature. The interpretation is given by

$$(2.79) \quad \mathcal{J}(f_{\vec{x}}) := \vec{x}, \quad \mathcal{J}(g_\rho) := \text{Adj}(\rho)$$

With this definition, the formal apparatus of the previous sections can be used with minor adaptations.

We say that  $G$  **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  $\gamma = \langle \vec{u}, \vec{v}, \vec{w} \rangle \in A^*$  such that  $\vec{z}$  possesses the decomposition  $\vec{z} = \gamma(\langle \varepsilon, \varepsilon \rangle) = \vec{u}\vec{v}\vec{w}$  and

$$(2.80) \quad \vec{y} = \gamma(\langle \vec{x}_0, \vec{x}_1 \rangle) = \vec{u}\vec{x}_0\vec{v}\vec{x}_1\vec{w}$$

$L(G)$  denotes the set of strings that can be generated in a finite number of steps. An alternative way to define this notion is to define the value of terms to be sets.

$$(2.81) \quad \iota_G(f s_0 \cdots s_{\Omega(f)-1}) := \left( \prod_{i < \Omega(f)} \iota_G(s_i) \right) \times A^* \cap \mathcal{J}(f)$$

For a zeroary mode  $f_{\vec{x}}$  we have

$$(2.82) \quad \iota_G(f_{\vec{x}}) = (1 \times A^*) \cap \{\vec{x}\} = \{\vec{x}\}$$

The other cases are similarly easy.

**Example 21.** We shall now give a presentation of the E-strings of the grammar from Example 18 using a string adjunction grammar. We put

$$(2.83) \quad S := \{\mathbf{0}=\mathbf{0}, \mathbf{0}=1, 1=\mathbf{0}, 1=1\}$$

The rules are as follows. Let  $\Lambda_1$  be the set of triples  $\langle \vec{u}x, \vec{v}, \vec{w} \rangle$  such that  $x$  is either  $/\mathbf{0}/$  or  $/1/$  and  $\vec{v}\vec{w}$  does not begin with  $/\mathbf{0}/$  or  $/1/$ . (This is equivalent with the following: (1)  $\vec{v} \neq \varepsilon$  and  $\vec{v}$  does not begin with  $/\mathbf{0}/$  or  $/1/$ , or (2)  $\vec{v} = \varepsilon$  and  $\vec{w} (!)$ )

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 end with /0/ or /1/,  $\vec{w}$  does not begin with /0/ or /1/, while  $\vec{v} \in \{0, 1\}^*$ .

$$\begin{aligned}
 \rho_0 &:= \langle \langle 0, \varepsilon \rangle, \Lambda_1 \rangle \\
 \rho_1 &:= \langle \langle 1, \varepsilon \rangle, \Lambda_1 \rangle \\
 \rho_2 &:= \langle \langle \varepsilon, 0 \rangle, \Lambda_1 \rangle \\
 (2.84) \quad \rho_3 &:= \langle \langle \varepsilon, 1 \rangle, \Lambda_1 \rangle \\
 \rho_4 &:= \langle \langle (, +0) \rangle, \Lambda_2 \rangle \\
 \rho_5 &:= \langle \langle (, +1) \rangle, \Lambda_2 \rangle \\
 R &:= \{\rho_0, \dots, \rho_5\}
 \end{aligned}$$

The signature is  $F := \{f_0, \dots, f_3, g_0, \dots, g_5\}$ , where the  $f_i$  are zeroary and the  $g_i$  are unary. Further,

$$\begin{aligned}
 \mathcal{J}(f_0) &:= \{0=0\} \\
 \mathcal{J}(f_1) &:= \{0=1\} \\
 (2.85) \quad \mathcal{J}(f_2) &:= \{1=0\} \\
 \mathcal{J}(f_3) &:= \{1=1\} \\
 \mathcal{J}(g_i) &:= \text{Adj}(\rho_i)
 \end{aligned}$$

Here is an example of a derivation:

$$\begin{aligned}
 f_1 & \quad 0=1 \\
 (2.86) \quad g_5 f_1 & \quad (0+1)=1 \\
 g_2 g_5 f_1 & \quad (0+10)=1 \\
 g_5 g_2 g_5 f_1 & \quad (0+(10+1))=1
 \end{aligned}$$

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

$$(2.87) \quad \gamma(\langle (, 1) \rangle) = \varepsilon \wedge (\wedge 0 \wedge +1) \wedge =1 = (0+1)=1$$

It may be checked that

$$\begin{aligned}
 & \iota_G(g_5 g_2 g_5 f_1) \\
 (2.88) \quad & = \{((\mathbf{0}\mathbf{0}+1)+1)=1, (\mathbf{0}\mathbf{0}+1)=(1+1), ((\mathbf{0}+1)+1\mathbf{0})=1, (\mathbf{0}+1)=(1\mathbf{0}+1), \\
 & ((\mathbf{0}+1)+1)=1\mathbf{0}, (\mathbf{0}+1)=(1+1\mathbf{0}), (\mathbf{0}\mathbf{0}+(1+1))=1, \mathbf{0}\mathbf{0}=(1+1)+1, \\
 & (\mathbf{0}+(1\mathbf{0}+1))=1, \mathbf{0}=(1\mathbf{0}+1)+1, (\mathbf{0}+(1+1))=1\mathbf{0}, \mathbf{0}=(\mathbf{0}+1)+1\mathbf{0}, \\
 & (\mathbf{0}\mathbf{0}+1)=(1+1), \mathbf{0}\mathbf{0}=(1+(1+1)), (\mathbf{0}+1\mathbf{0})=(1+1), \mathbf{0}=(1\mathbf{0}+(1+1)), \\
 & (\mathbf{0}+1)=(1\mathbf{0}+1), \mathbf{0}=(1+(1\mathbf{0}+1))\}
 \end{aligned}$$



**Example 22.** (Cf. Example 7.) We give another example: boolean logic in Polish Notation. The alphabet is  $\text{:bool:} = \{\wedge, \vee, \neg, p, \mathbf{0}, 1\}$ . A term in Polish Notation is either  $/p/$  followed by an index (a sequence of  $/\mathbf{0}/$  and  $/1/$ ) or it is a function symbol  $f$  ( $\neg$ ,  $\wedge$  or  $\vee$ ) followed by  $\Omega(f)$  many terms. The formation rules using adjunction grammars are as follows. The set of start strings is  $S := \{p\}$ . The rules are

$$\begin{aligned}
 (2.89) \quad R := & \{ \langle \langle \mathbf{0}, \varepsilon \rangle, \langle A^* \cdot p, \varepsilon, A^* \rangle \rangle, \\
 & \langle \langle 1, \varepsilon \rangle, \langle A^* \cdot p, \varepsilon, A^* \rangle \rangle, \\
 & \langle \langle \neg, \varepsilon \rangle, \langle A^*, (p|\wedge|\vee|\neg) \cdot A^*, \varepsilon \rangle \rangle, \\
 & \langle \langle \wedge p, \varepsilon \rangle, \langle A^*, (p|\wedge|\vee|\neg) \cdot A^*, \varepsilon \rangle \rangle, \\
 & \langle \langle \vee p, \varepsilon \rangle, \langle A^*, (p|\wedge|\vee|\neg) \cdot A^*, \varepsilon \rangle \rangle \}
 \end{aligned}$$

Using the Exercises 4 and 5 we can see that this preserves termhood: the sum of the elements added in the string is 0, and the sum of the prefixes is positive. The original Polish Notation had no room for indices, but they pose no problem here. It is easy to verify that any string in Polish Notation is derivable in this grammar.



It is easy to generalise the previous example to Polish Notation in general (see the exercises). Furthermore, I describe in the exercises how one can derive an adjunction grammar for bracketed notation as well.

In the remainder of this section I shall describe two variants of adjunction grammars that have been discussed in the literature.

**Definition 2.24** A locale  $\Lambda$  is **factored** if there are sets  $S \subseteq A^* \times A^*$  and  $C \subseteq A^*$  such that  $\Lambda = \{\langle \vec{u}, \vec{v}, \vec{w} \rangle : \langle \vec{u}, \vec{w} \rangle \in S, \vec{v} \in C\}$ . A rule is **factored** if its locale is. A **contextual grammar** is a string adjunction grammar in which every rule is factored.

See [Martín-Vide and Păun, 1998] for an overview.

The most popular variant of adjunction grammars are however the tree adjunction grammars (TAGs). These grammars can be explained by a method of coding trees into strings. To make matters even simpler, we simply declare certain strings to *be* trees. Let  $N$  be a set, the set of **nonterminal labels**. Then  $N$ -trees over the alphabet  $A$  are strings from  $A \cup N \cup \{ (, ), \downarrow \}$ . (a)  $x \in A^*$  is an  $N$ -tree; (b) if  $\vec{u}_i, i < n$ , are  $N$ -trees, and  $X \in N$ , then  $(X\vec{u}_0\vec{u}_1 \cdots \vec{u}_{n-1}X)$  and  $\downarrow (X\downarrow\vec{u}_0\vec{u}_1 \cdots \vec{u}_{n-1}\downarrow X)\downarrow$  is an  $N$ -tree. The adjunction rules have the following form. Let  $\langle \vec{x}_0, \vec{x}_1 \rangle$  be a pair such that  $\vec{x}_0 = \downarrow (X\downarrow \cdots, \vec{x}_1 = \cdots \downarrow X)\downarrow$  and  $\vec{x}_0\vec{x}_1$  is an  $N$ -tree. Such pair shall be called an  **$N$ -adjunction tree**. Given this tree, let

$$(2.90) \quad \Lambda := \{\langle \vec{u}, \vec{v}, \vec{w} \rangle : \vec{u}\vec{v}\vec{w} \text{ is an } N\text{-tree, } \vec{v} = (X \cdots X)\}$$

The pair  $\langle \langle \vec{x}_0, \vec{x}_1 \rangle, \Lambda \rangle$  is called a **tree adjunction rule**. Notice that the category  $X$  of the adjunction string must match the  $X$  in the locale. Also, the presence of  $\downarrow$  blocks adjunction at a node. (The symbol  $\downarrow$  is not needed to code the tree structure; its sole purpose was to restrict adjunction.) There are many variants of TAGs. We have picked the most common form for comparison. The language generated by a TAG  $G$  is however not the string language; rather it is the language of yields. This is defined as follows. Put

$$(2.91) \quad h^\downarrow(x) := \begin{cases} \varepsilon & \text{if } x \in N \cup \{ (, ), \downarrow \} \\ x & \text{else} \end{cases}$$

$$h^\downarrow(x_0x_1 \cdots x_{n-1}) := h^\downarrow(x_0)h^\downarrow(x_1) \cdots h^\downarrow(x_{n-1})$$

Then  $L_Y(G) := h^\downarrow[L(G)]$  is the language of yields, by definition the language generated by  $G$ .

**Exercise 17.** Verify that the grammars from Examples 21 and 22 are contextual grammars.

**Exercise 18.** Let  $\Omega$  be an arbitrary signature. Write an adjunction grammar for all terms in Polish Notation in this signature.

**Exercise 19.** Let  $\Omega$  be an arbitrary signature. Terms in this signature are now written as follows. If  $f$  is binary, and  $s$  and  $t$  terms, then  $(\hat{\ }s\hat{\ }f\hat{\ }t\hat{\ })$  is a term. If  $f$  is unary then then  $f\hat{\ }(\hat{\ }s\hat{\ })$  is a term. If  $f$  is ternary and higher order, terms are of the form  $f\hat{\ }(\hat{\ }s_0\hat{\ },\hat{\ },\hat{\ } \cdots \hat{\ },\hat{\ } s_{\Omega(f)-1})$  is a term. Use the previous exercise to derive an adjunction grammar for this language.

## 2.5 Syntactic Structure

Contemporary linguistics insists that what matters is not the string that we see but rather its *structure*. Structure usually means tree structure. It has been stressed by Chomsky that rules operate on constituents and not on strings. Moreover, Transformational Grammar uses representations that contain the structure in them. Formally, however, it is not clear whether the structure needs to be represented. In this section I shall discuss a popular way to encode structure into the string. Moreover, we shall investigate to what extent a context free language determines the grammar from which it is generated.

Let us take a look at CFGs and tree structures. Given a string  $\vec{x}$  and a grammar  $G$  that generates it,  $G$  assigns a structure to  $\vec{x}$  through a term in the following way. Assume a term  $t$  for the string  $\vec{x}$ . Then  $t = f_\rho(s_0, \dots, s_{n-1})$ , where  $n = \Omega(f_\rho)$ .

$$(2.92) \quad \rho = A \rightarrow \vec{x}_0 B_0 \vec{x}_1 B_1 \cdots B_{n-1} \vec{x}_n$$

If  $n = 0$ ,  $\rho = A \rightarrow \vec{x}_0$ , and we just let the tree consist of two nodes, one with label  $\vec{x}_0$ , and a preterminal with label  $A$ . In general, we create a daughter for each  $B_i$  and attach the tree for  $s_i$  there, and a daughter for every nonempty  $\vec{x}_i$  whose label will be  $\vec{x}_i$  (we avoid positing empty words).

We can code the derivation into the string. This is done by switching to a grammar that distributes brackets.  $G^b$ . This grammar is defined 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

$$(2.93) \quad \rho^b := X \rightarrow ({}_X \vec{Y})_X$$

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

$$(2.94) \quad R^b := \{\rho^b : \rho \in R\}$$

Let  $\vec{x}$  be a string. Each term  $t$  of  $\vec{x}$  can be mapped to a term  $t^b$ , which is defined by replacing every occurrence of  $f_\rho$  by an occurrence of  $f_{\rho^b}$ , for every occurrence and every  $\rho$ . Thus mapping  $t$  into a term  $t^b$  of the bracketed grammar we find the string  $\vec{x}_t^b$ , which contains a record of  $t$ .  $\vec{x}^b$  is obtained by  $\vec{x}_t^b$  by deleting the brackets and the category symbols. More exactly, define a map  $d$  as follows.

$$(2.95) \quad d(a) := \begin{cases} \varepsilon & \text{if for some } X: a = ({}_X \text{ or } a = )_X \\ a & \text{else} \end{cases}$$

$$d(x_0x_1 \cdots x_{n-1}) := d(x_0)d(x_1) \cdots d(x_{n-1})$$

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 23.** Let  $G = \langle \text{blet}; \{E, A, B\}, E, R \rangle$  where  $R$  contains the following rules:

$$(2.96) \quad \begin{aligned} E &\rightarrow AB \mid BA \mid EE \\ A &\rightarrow AE \mid EA \mid a \\ B &\rightarrow BE \mid EB \mid b \end{aligned}$$

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

$$(2.97) \quad \begin{aligned} E &\rightarrow (EAB)_E \mid (EBA)_E \mid (EEE)_E \\ A &\rightarrow (AAE)_A \mid (AEA)_A \mid (Aa)_A \\ B &\rightarrow (BBE)_B \mid (BEB)_B \mid (Bb)_B \end{aligned}$$

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:

$$(2.98) \quad \begin{aligned} &(E(E(Aa)_A(Bb)_B)_E(E(Aa)_A(Bb)_B)_E)_E \\ &(E(Aa)_A(B(E(Bb)_B(Aa)_A)_E(Bb)_B)_B)_E \end{aligned}$$



Erasing the brackets returns the original string. The derivation  $/E/, /EE/, /EAB/, /ABAB/$  on the other hand yields the first string again.  $\odot$

**Proposition 2.25**  $G^b$  is unambiguous.

The proof is straightforward. It rests on the usual bracket count of embeddings.

Notice however that the structure of  $\vec{x}$  is a derived notion and the bracketed string just a theoretical construct. The structure 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. We shall now discuss the notion of constituent occurrence without adding brackets. Recall the definition of an occurrence from Definition 2.11. Given a grammar  $G$  and a term  $t$  we can assign constituent occurrences of substrings in a straightforward way. Choose a subterm occurrence  $s$  and decompose  $t$  into  $t = t'(s)$ . This means that  $t'(x_0)$  is a term with one free variable and it defines a function  $\iota_G(t'(x_0)) : \vec{x} \mapsto \vec{u}\vec{x}\vec{v}$ . This means that  $\langle \vec{u}, \vec{v} \rangle$  is a 1-context, and the substring that occurs in  $t$  is  $\iota_G(s)$ . For a constant term  $t$ ,  $\text{occ}(\vec{y}, t)$  is the set of occurrences of  $\vec{y}$  in  $\iota_G(t)$ .

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 2.26** Let  $G$  be a CFG,  $t$  an  $A$ -analysis of the string  $\vec{x}$  and  $C$  an occurrence of  $\vec{y}$  in  $\vec{x}$ . If  $C \in \text{occ}(\vec{y}, t)$  then  $C$  is said to be a **constituent occurrence of  $\vec{y}$  in  $\vec{x}$  under the analysis  $t$** . If  $C \notin \text{occ}(\vec{y}, t)$ , the occurrence is said to be **accidental  $A$ -occurrence** if  $\vec{y} \in L_A(G)$ .  $G$  is **transparent** if no constituent has an accidental occurrence in a string of  $L(G)$ . A language is **transparent** if it has a transparent grammar.

Notice that we look at occurrences under a given analysis term  $t$ . A given string  $\vec{x}$  can in principle have several analyses. Suppose that a context free language  $L$  is transparent. Then given a string  $\vec{x} \in L$  we know that every substring occurrence of  $\vec{x}$  which is in  $L$  also is a constituent occurrence under every analysis. Thus any context free grammar will assign the same constituent tree to  $\vec{x}$ . This is very useful for languages of analysis terms, because we need to know that they are structurally unique. This is the case for  $\text{Tm}_\Omega(V)$ , as the next theorem asserts.

**Proposition 2.27** *The language  $\text{Tm}_\Omega(V)$  is transparent.*

**Proof.** (For notation and facts see the Exercises 4 and 5.) Let  $s$  and  $t$  be terms, and  $C = \langle \vec{u}, \vec{v} \rangle$  an occurrence of  $s$  in  $t$ . We shall show that  $s$  is actually a subterm occurrence of  $t$  by induction on  $t$ . For either (a)  $\vec{u} = \varepsilon$  or (b)  $\vec{u} \neq \varepsilon$ . If (a) is the case then  $\vec{v} = \varepsilon$ , that is,  $s = t$ , or else  $s$  is a proper prefix. This cannot be, since this would imply  $\gamma(s) \geq 0$ . Now in case (b) there is an  $i$  such that the named occurrence begins in  $t_i$ . (Case 1) The occurrence is contained in  $t_i$ , that is,  $t_i = \vec{x}s\vec{y}$  for some  $\vec{x}$  and  $\vec{y}$ . Then we are done by inductive hypothesis. (Case 2)  $s$  overlaps with  $t_i$ . Then we have  $\vec{x}, \vec{y}$  and  $\vec{z}$  all nonzero such that  $t_i = \vec{x}\vec{y}$  and  $s = \vec{y}\vec{z}$ . Now note that since  $-1 = \gamma(t_i) = \gamma(\vec{x}) + \gamma(\vec{y})$  and  $\gamma(\vec{x}) \geq 0$  (since  $t_i$  is a term) we must have  $\gamma(\vec{y}) < 0$ . but then  $s$  is not a term since  $\gamma(\vec{y}) \geq 0$  if  $\vec{y}$  is a proper prefix. So this case does not arise and we are done.  $\square$

Every constituent occurrence in  $\vec{x}$  under  $t$  corresponds to a subterm occurrence in  $t$ . We use this for the following definition. A term is simple if has no nontrivial subterms.

**Definition 2.28** *Let  $G$  be a CFG,  $t$  an  $A$ -analysis of the string  $\vec{x}$  and  $C$  an occurrence of a letter  $\vec{y}$  in  $\vec{x}$ .  $C$  is **syncategorematic** if the term to which it belongs is not simple. A substring occurrence is syncategorematic if every letter is syncategorematic and belongs to the same subterm.  $G$  is in **standard form** if no string has syncategorematic occurrences.*

A CFG is standard if the functions are terms. This definition can easily be generalised. For a CFG, being in standard form means that the right hand side of a rule cannot contain both a nonterminal and a terminal letter. For example, the standard formulation of regular grammar is that they have rules of the form  $A \rightarrow xB$  or  $A \rightarrow x$ . Such grammars are not standard. It is easy to convert a CFG into standard form. However, notice that this changes the language of the grammar, since for us the language contains all constituents.

**Example 24.** We continue Example 23 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/.  $\odot$

**Proposition 2.29** *Let  $G$  be a CFG,  $\vec{x} \in L(G)$  and  $t$  an analysis term. Fix a constituent occurrence of  $\vec{y}$  in  $\vec{x}$  under  $t$ . 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  $\vec{x}, \vec{y} \in S_i$  then 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 25.** The language of equations. In the form presented in Example 18 on Page 51. This grammar introduces /=/, the operation symbols and the brackets in a syncategorematic way. It can be reformulated as follows. The original rule set is

$$\begin{aligned} E &\rightarrow T=T \\ (2.99) \quad T &\rightarrow (T+T) \mid (T-T) \mid B \\ B &\rightarrow B0 \mid B1 \mid 0 \mid 1 \end{aligned}$$

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

$$\begin{aligned} O &\rightarrow ( \\ C &\rightarrow ) \\ (2.100) \quad P &\rightarrow + \\ M &\rightarrow - \\ Q &\rightarrow = \end{aligned}$$

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

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

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:

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



Notice that the grammars (2.100) and (2.102) are not only different grammars; they in fact generate different languages. For example, the string /(/ is in the language of (2.102), but not in (2.100). This is a consequence of the fact that we defined  $L(G)$  to contain *all* constituents of  $G$ , not just the sentences.

Let us now turn to the idea of recovering the grammar from the set of strings. We start with the assumption that our language is generated by a context free grammar. This means that constituents are strings, and that a string is a part of another string only if it is a subword. The standard substitution method starts with the language  $L$ , and establishes for every  $\vec{x} \in L$  the set of contexts:

$$(2.103) \quad \text{cnt}_L(\vec{x}) := \{ \langle \vec{u}, \vec{v} \rangle : \vec{u}\vec{x}\vec{v} \in L \}$$

The syntactic classes are the context sets so obtained. We present an example first.

**Example 26.** Continuing Example 11. The language is generated by  $u$ , defined by

$$\begin{aligned}
 (2.104) \quad & t := \text{Alex}_\perp \mid \text{Pete}_\perp \mid \text{Mary}_\perp \\
 & u := t(\text{and}_\perp t)^*(\text{sing}_\perp \mid \text{run}_\perp \mid \text{sings}_\perp \mid \text{runs}_\perp)
 \end{aligned}$$

We consider words as units together with a following blank. (This makes the calculations easier.) The context sets are as follows. Here is a more succinct definition of the language:

$$(2.105) \quad \begin{aligned} a &:= \text{and}_\perp \\ v &:= \text{sings}_\perp \mid \text{runs}_\perp \\ w &:= \text{sing}_\perp \mid \text{run}_\perp \\ u &= tv \mid t(at)^+w \end{aligned}$$

It turns out that the syntactic classes are the following:  $v, tv, w, a, t, ta, at, tat, atatat, atw, tatw$ . These are more classes and more constituents than the were present in the original grammar even if we massaged the syncategorematic occurrences away.  $\odot$

The exercises give one more example. The problem with the substitution method is that there is no way of telling whether an occurrence is accidental or not. Consequently, the method will return context sets that are the sets of non-constituent occurrences. In fact, we may end up with infinitely many context sets (see the exercises). And this is not because of the finiteness of the data: even if we had all data in our hands, the grammar is still underdetermined. Thus, there is some art involved in establishing the subset of constituent occurrences. This set can be different from the one for the original grammar. However, in the absence of decisive evidence this is the best one can do.

Under certain circumstances we can know in advance that the set of nonterminals is going to be finite. A particular case is provided by *primitive languages*.

**Definition 2.30** *A language is called **primitive** if every substitution class contains a string of length 1, that is, consisting of a single letter.*

Evidently, since the alphabet is finite, there are finitely many substitution classes. This does not guarantee the uniqueness of the solution (see the exercises) but it narrows the choice considerably.

The language defined above is not primitive. 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.

Primitive 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

$$(2.106) \quad \begin{array}{ll} N_a \rightarrow a & a \in A \\ N_a \rightarrow N_{c_0} N_{c_1} \cdots N_{c_n} & c_0 c_1 \cdots c_n \in [a]_L \end{array}$$

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


We shall finally turn to the abstract case.

**Definition 2.31** *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 orthographically definite if and only if  $s(v)$  is. They are **intersubstitutable**, in symbols  $u \approx_G v$ , if and only if they are categorially equivalent and  $\iota(s(u)) \in L(G)$  if and only if  $\iota(s(v)) \in L(G)$ .*

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

$$(2.107) \quad \begin{array}{l} S \rightarrow ATB \\ T \rightarrow x \mid xT \\ Ax \rightarrow xA \\ AB \rightarrow y \end{array}$$

In a derivation, first  $/A/$  is generated to the left of the string. However, when the last rule applies,  $/A/$  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 20.** Show that the substitution classes of a context free grammar (construed 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  $Y \in N$ :  $Y \Rightarrow^* \vec{x}$  if and only if  $Y \in P$ .

**Exercise 21.** Apply the method of context sets to grammar  $C_1$  of Example 9. Show that the grammar that this gives is  $C_2$  (also from Example refstrings)! Show that the language generated by either grammar is primitive.

**Exercise 22.** Let  $G$  consist of the rules  $S \rightarrow ab \mid aSb$ . Establish the context sets of all substrings and show that there are infinitely many of them. Show that infinitely many context free grammars can be postulated on the basis of these sets.

**Exercise 23.** Let  $G$  be a context free grammar. Try to establish an inductive definition of  $\text{occ}(\vec{y}, t)$ . *Hint.* This definition will have to be inductive in the length of  $\vec{y}$  and  $t$ .

## 2.6 The Principle of Preservation

We have seen that the effect of substitution is unpredictable unless restrictions are placed on the nature of the string functions. We propose here two principles that simplify the situation. In the most ideal case, functions are only able to change a string except by appending material to its left or right. If we required this we get something slightly more general than context free grammars. To get some more freedom we propose that grammars do not operate on the set  $A^*$  but on some slightly more general set of exponents, which we equate with  $(A^*)^m$  for some  $m$ , or perhaps  $\bigcup_{m \in \mathbb{N}} (A^*)^m$ , as proposed in [Kracht, 2003].

**Principle 2 (Structure)** *Exponents are sequences of strings.*

This is a heavy restriction but it still allows substantial freedom, more than is immediately apparent. 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 I proposed above, sequences thereof. This latter qualification is important. Consider the following principle.

**Principle 3 (Structure Preservation)** *A rule may not break any string of the exponent or delete any parts of it.*

This can be formalised as follows. The interpretation of a function symbol  $f$  is a function from  $\Omega(f)$  many  $m$  tuples to a single  $m$ -tuple of strings. So,  $\iota(f) = \langle t_0, t_1, \dots, t_{\Omega(f)-1} \rangle$ , where the  $t_i$  are terms in  $m < \Omega(f)$  variables which are polynomial functions in the string algebra

$$(2.108) \quad \langle A^*, \varepsilon, \wedge \rangle$$

over the signature  $\Omega_{\wedge} := \{\langle \varepsilon, 0 \rangle, \langle \wedge, 2 \rangle\}$ . This means further that  $t_i$  may use variables, constants for letters of  $A$  and for the empty string, and concatenation.

What these principle rule out is deletion of any kind; they also rule out breaking a constituent. 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 strings. 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, even in the case of these languages it is not entirely clear that the approach does not work; for these languages do not break embedded clauses either. This needs further work.

We have so far only spoken about breaking or deleting strings. The next principle talks about nonlexical rules.

**Principle 4 (Syncategorematicity Prohibition)** *A rule may not add any occurrence of a given symbol.*



Again, this can be formalised by saying that the interpretation of functions uses only definable term functions in  $\Omega_{\omega}$ , not polynomials. This allows for complete reduplication (as in Malay), and it also allows for partial reduplication (modulo a regular relation), 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

$$(2.109) \quad r(x) = x \hat{\ } x$$

or by partial concatenation, using the function

$$(2.110) \quad c(x, y) = \begin{cases} x \hat{\ } y & \text{if } x = y \\ \text{undefined} & \text{else} \end{cases}$$

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.

Another hard case to treat is the so-called **tnesis**. This is the coordination of parts that are not words by themselves. For example, in German we have the words /Urfeind/ and /Erzfeind/, both formed from /Feind/ ‘enemy’ and a prefix /Ur/ ‘since very long ago’ and /Erz/ ‘arch-’. What is striking is that while neither prefix can be on its own, it is possible to say

$$(2.111) \quad \text{Ur- und Erzfeind}$$

Similarly, verbal prefixes can be separated

$$(2.112) \quad \text{auf- und abladen} \quad \text{‘load and unload’}$$

<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 properties 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.

Tmesis can be applied at the juncture of compounds and with certain prefixes. It is in particular not free to apply to any morphological part of the word. A proper formulation of tmesis under the conditions just sketched is not impossible but requires great care.

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 to  $\vec{x}$ . Again, requiring this we do not so much restrict *what* can be done but rather *how* it can be done. Yet, there is a problem with this requirement, and it runs as follows. We practically assume that also bound forms are part of the language; that is, the plural /s/ of English, even though it cannot occur on its own, is part of the English language. However, that might just be an artefact of the requirement that only words are free forms; and we may say that the language consists of more than just the free forms. The semantics of the plural on the other hand is unproblematic or at least not more problematic than that of any other item.

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). On the other we have ideographic systems like Chinese, which make a single letter correspond (again more or less) to a morpheme. Chinese presents a good example of the predicament we are facing: if we base our analysis on the *sounds* 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 50,000 ‘letters’. (The Chinese Standard Interchange Code, the most comprehensive of the lists, has close to 50,000 characters.) 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 them from the analysis.<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,

---

<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) and ‘shān’ in Chinese, but they are also arranged according to Chinese syntax.

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 *morph*, or *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 morph(eme) makes perfect sense, because once we add the meaning the alphabetic characters are in fact *not* the most basic elements, but the morph(eme)s.

In stratificational 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'cuk, 1993 2000]. In our view the various levels are mostly epiphenomenal and can be reconstructed on the basis of the language (as a set) itself. I shall briefly discuss the reconstruction of levels in Section 3.7.

Even if all this is 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 that is 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, or consonant gradation in Sami ([Svenonius, 2007]). 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 respective genitives, /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 in isolation, so we add a word boundary marker to its left and right:  $[\#_B a: d \#]$  and  $[\#_B a: t \#]$ . 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 a *regular relation* (see [Kracht, 2003] for definitions and discussion). To account for German devoicing, we assume that at 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.

Let us briefly touch on the question of *c*-languages. If one wishes to include categories into the language then the Principle of Preservation loses some of its bite. It would namely be possible to introduce material into the category part where it is invisible to the principles formulated above. I assume therefore that when categories are added they cannot introduce a finer distinction than already present in the functions:

**Principle 5 (Categorial Granularity)** *For a  $c$ -grammar  $G$  and the associated string grammar  $H$ , if  $\langle \vec{x}, c \rangle \in L(G)$ , and  $\vec{y} \sim_H \vec{x}$  then also  $\langle \vec{y}, c \rangle \in L(G)$ .*

Thus, the set of categories cannot differentiate the exponents in a finer way than the string functions. The way this is phrased makes the principle somewhat circular. But you need to recall that the string categories are derived from the string functions and ultimately from the language itself. Thus, bringing in an extra set  $C$  of categories really is to serve the purpose of explicitly coding the categorial facts rather than bringing back a lost dimension. However, I add that adding categories even with the Granularity Principle brings in extra power.

**Exercise 24.** German nouns are written with an initial upper case letter. However, in compounds only the first letter is in upper case. For example, /Auto/ ‘car’ and /Bahn/ ‘way’ result in the compound /Autobahn/ ‘highway’. (Observe similarly /Erzfeind/ in the example above.) Propose a solution to this. *Hint.* There are two solutions. One uses the regular relations, the other proposes several forms for the same word.



# Chapter 3

## Compositionality

THE principle of compositionality will be introduced in this chapter: it concerns the relationship of strings with their meanings. To be able to formulate it properly, we shall have to introduce interpreted languages and grammars for them.

### 3.1 Compositionality

Let us begin with some exegetical remarks concerning the notion of compositionality. Here is what I regard as a standard definition of compositionality.

The meaning of a complex expression is a function of the meanings of its parts and the mode of composition by which it has been obtained.

Almost every word of this definition is in need of explanation. We begin with the subject of the sentence: *the meaning of an expression*. To use this expression here means that there first of all *are* expressions and meanings; and that expressions have meanings. Immediately we start to ask ourselves what expressions *are* and what meanings *are*. Since meanings are attributed to expressions, I take this to say that whatever expressions are, they must be part of the language to begin with. Thus, strictly speaking, expressions must be strings. However, we have settled the question differently in Section 2.5: expressions are *sequences* of strings. More-

over, they must be sequences of strings of which we know what their meaning is. This is implicit in the use of the definite determiner “the meaning of an expression”. The use of the definite determiner “the” is somewhat troublesome: it may mean that an expression has one and only one meaning; it may also mean that its meaning is not arbitrary. If taken in the first sense expressions are unambiguous. I take this to be incorrect and not the way in which “the” is to be understood here (see also the discussion in Section 3.5). Rather, I wish to plead that we interpret this as follows: given that we are under way to investigate *some* meaning of an expression, which is one of the many that it may have, but we have fixed that one as opposed to others, we have a recipe to get *this* meaning from whatever the components mean. Thus, the definite determiner points to an implicitly made choice. I defer a definition of what meanings are. So far we know this much: there are expressions (sequences of strings) and meanings; a language consists of a relation between the two. This is the original idea laid out in [Saussure, 1967].

One word still remains to be discussed: *complex*. To say whether an expression is simple or complex cannot be determined intrinsically; in fact, ‘complex’ here means the following. We have a grammar  $G$  of the expressions. An expression is **simple** if it is the value of a simple term; and an expression is **complex** if it is the value of a complex term. It turns out that one and the same expression can both be simple and complex; this is the case with idioms, for example. But it is also the case with false idioms such as /caterpillar/. This expression is both simple and complex, at least if we use a simple compounding grammar of English. Notice that so far the grammar is just a context free grammar for tuples of strings and knows nothing about the meaning. To make sense of the above definition, however, we must assume that the grammar also handles meanings together with expressions. For we wish to say, for example, that idioms are simple. For although as expressions they are complex, their meaning is not derived from the parts their expressions have.

We are thus led to assume that the definition of compositionality talks about *languages as relations between expressions and meanings and grammars that generate such relations from a given finite set*. It is this type of language and grammar that we shall look at in detail in this chapter. We call them *interpreted languages* and *interpreted grammars*. To finish explicating the definition, let us assume that we have such a grammar that generates not just expressions but pairs of expressions and meanings. Such pairs we call from now on *signs*. A sign is thus a pair  $\sigma = \langle e, m \rangle$ , where  $e$  is the *exponent* of  $\sigma$  and  $m$  the *meaning*. While it



cannot be said that in a given language a given expression has just one meaning and a given meaning has just one expression, it is true by definition that a given sign has exactly one exponent and one meaning. It thus seems that it is more appropriate to exchange ‘expression’ in the above definition by ‘sign’. It therefore reads as follows.

The meaning of a complex sign is a function of the meanings of its parts and the syntactic rule by which it has been composed.

Let us try to understand this definition further. A grammar generates signs; it starts with a lexicon, which we may take to be a finite list of signs. In addition it has some functions to generate signs from signs, in the same way as a string grammar generates strings from strings.

A sign  $\sigma$  is simple if and only if it is the value of a simple term; it is complex if and only if it is the value of a composite term. A given sign can be both simple and complex. The previous problems have now disappeared. An idiom for example is a sign that is simple but not complex, because its meaning is not obtainable in the grammar in a regular way. So, the definition begins by assuming that we have a grammar  $G$  and a sign  $\sigma$ . Furthermore, we assume that there is a term  $t(\vec{x})$  and signs  $\sigma, \sigma_0, \dots, \sigma_{n-1}$  such that

$$(3.1) \quad \sigma = t(\sigma_0, \dots, \sigma_{n-1})$$

In that case assume that  $\sigma_i = \langle e_i, m_i \rangle$  and  $\sigma = \langle e, m \rangle$ . Then

$$(3.2) \quad m = F(t, m_0, \dots, m_{n-1})$$

for some  $F$  that depends only on  $G$ . We can without further ado write  $t^\mu$  for the function  $F(t, -, \dots, -)$ . Then the previous means that

$$(3.3) \quad m = t^\mu(m_0, \dots, m_{n-1})$$

It follows by a simple argument (induction on the length of  $t$ ) that it is enough to require (3.3) for  $t$  a basic function of  $G$ .

At last we need to clarify the notion of a *mode of composition*. First of all, we use the same terminology as in the preceding chapter. We assume that we have a finite set  $F$  of function symbols forming a signature  $\langle F, \Omega \rangle$  together with  $\Omega$ . As

we saw above, for each  $f \in F$  there is an  $f^\mu$  satisfying (3.3). This is the meaning function; there also is a function  $f^\varepsilon$  such that

$$(3.4) \quad e = f^\varepsilon(\sigma_0, \dots, \sigma_{\Omega(f)-1})$$

We shall see later that one would ideally impose some restrictions on  $f^\varepsilon$ . Crucially, we may understand *mode* as referring just to  $f$ , or as referring in fact to  $f^\varepsilon$ . Suppose for example that we have the following language  $L$ .

$$(3.5) \quad L = \{\langle a, 0 \rangle, \langle b, 1 \rangle, \langle ab, 2 \rangle, \langle ab, 3 \rangle\}$$

Assume that  $/ab/$  is to be considered complex. If we understand a mode to be a syntactic function this language cannot be compositional, for there is only one function to compose  $/a/$  and  $/b/$ .<sup>1</sup> To make this even more precise: we shall assume that what counts in the specific case is not the function as a whole but rather what it does to the specific elements at hand. That is to say that we can also define the following function:

$$(3.6) \quad f(x, y) := \begin{cases} x \frown y & \text{if } x = a \text{ and } y = b \\ y \frown y & \text{if } x = aa \text{ and } y = b \\ \text{undefined} & \text{else} \end{cases}$$

This is a different function, but on the strings of the language it shows no difference to plain concatenation. We say therefore that  $f^\varepsilon$  and  $g^\varepsilon$  *count as the same mode* exactly when  $f^\varepsilon(\vec{\sigma}) = g^\varepsilon(\vec{\sigma})$ . There are languages which satisfy compositionality even with this strict identity of modes; many computer languages are of that form. There simply is only one way to combine two constituents semantically; the surface syntax may be flexible (allowing the use of brackets, for example), but this is just a means of identifying the constituents. However, semantically, there is just one way to combine two meanings. Natural languages are quite different in this respect. Many expressions are naturally ambiguous.

Let us now settle down on the final definition of compositionality:

A language is **compositional** if there is a grammar  $G$  based on a signature  $\langle F, \Omega \rangle$  if for each  $f \in F$  there are functions  $f^\mu$  such that if  $\sigma = \langle e, m \rangle$  and  $\sigma_i = \langle e_i, m_i \rangle$ ,  $i < \Omega(f)$ , are signs such that

$$(3.7) \quad \sigma = f(\sigma_0, \dots, \sigma_{\Omega(f)-1})$$

<sup>1</sup>Well, there are two:  $f(x, y) := x \frown y$   $g(x, y) := y \frown x$ . But this can be handled by constructing a more complex example.

and  $g$  is the same mode as  $f$  then

$$(3.8) \quad m = g^\mu(m_0, \dots, m_{\Omega(f)-1})$$

Three notions of sameness come to mind: (a)  $f = g$  (symbolic identity), (b)  $f^\varepsilon = g^\varepsilon$  (extensional identity), and (c)  $f^\varepsilon(\vec{\sigma}) = g^\varepsilon(\vec{\sigma})$  (casewise identity). Option (c) is the least strict on the functions (and therefore induces the strictest condition on compositionality); in that case, any two functions which are defined at all on the input (and return the output string) are the same for the purpose of the definition.

A last point to mention is that strings may have *categories*. In that case we may further refine the notion of identity, allowing functions to depend on the categories of the arguments. I shall discuss the ramifications of this option below.

I shall now review some alternative definitions of compositionality. First, there is a tradition to use a more elaborate structure than the string, namely a tree structure defined over the string. In fact there are several such structures, and it is one of them that is actually interpreted, namely LF. The meaning of a particular LF is actually independent of the way in which it was obtained; however, as it has internal structure, its meaning can be obtained with reference to that structure. I shall return to the question of the viability of this proposal in Section 5.4. Here I just notice that to safeguard themselves from a different interpretation of compositionality some people have named the concept used here **rule-to-rule compositionality**, or **direct compositionality** (see the volume [Barker and Jacobson, 2007]). I shall not follow that usage partly because I think that the alternative notions are too weak to yield interesting results.

More interesting therefore are definitions that are more restrictive than the one given here. [Szabó, 2000] gives the following definition.

The meaning of a complex expression is determined by the meaning of its constituents and by its structure.

In his discussion, Szabó focuses mainly on the word ‘determines’. The idea is that ‘determines’ refers to some causal connection. Thus a language that uses just any function is not good enough. Some essential link must exist between the structure and the meaning. Thus, Szabó claims, we are led to assume that in order for the meaning to be determined by the structure, meanings must be structured and there must be a kind of structural parallel between syntax and semantics. The structure

in meaning is language independent, so this would among other imply a certain similarity between all human languages. We have chosen not to go that way. One reason for our choice is that the structure of meanings is something that we believe is too poorly understood to give insightful results at this point; thus, I am not arguing that meanings are not structured, I am only saying that the actual structure they have—whatever it may be—is very hard to determine. The recent discussion in [King, 2007] I do not find very revealing in this connection and too much language bound. Should it turn out that meanings *are* structured our approach is nevertheless not invalid; there will then be more conditions on syntactic structure. I think that one need not believe in structured meanings in order to establish a difference between just any kind of meaning composition function and one that is ‘good’, that is, ‘compositional’. I shall return to the question of natural meaning functions in the next chapter.

Another notion of compositionality is that of [Hodges, 2001]. In essence, the definitions are the same as the ones given here; there are however some technical differences that need to be pointed out. The main difference is that Hodges assumes that meanings are given to an expression through a function; thus an expression always has a unique meaning. This simplifies the technical apparatus and works well for artificial languages, but for natural language this is actually a problematic assumption. Notice that it eliminates ambiguity. Words such as /bank/ or /crane/ will not be considered ambiguous by the grammar. Moreover, the semantic functions  $f^\mu$  will operate on the total meaning. This means the following: an adjective such as /big/ does not simply operate on the different meanings of /crane/ independently; rather, it operates on the combined meaning of the two. Let us make that concrete. /crane/ either means a type of birds—call this meaning  $\text{crane}'_b$ —, or a type of machines—call the meaning  $\text{crane}'_m$ . The meaning function now associates with it the concept  $\text{crane}'_b \vee \text{crane}'_m$ , which is true of  $x$  if and only if  $x$  is either a bird crane or a machine crane. The meaning  $\text{big}'$  of /big/ on its part takes the whole concept and forms the concept of being-a-big-crane. Evidently, big bird cranes are far smaller than big machine cranes, so we expect the idea of a big bird-or-machine-crane to be different from both.

We may try to save the theory by proposing that the meaning of an ambiguous item is the set of different meanings it has otherwise. Thus, we assign to /crane/ the meaning  $\{\text{crane}'_b, \text{crane}'_m\}$ . This opens problems of its own. For example, an adjective will now apply to a set of what we otherwise would call meanings. How does it apply to such a set? We will have to say that it applies to each member indi-

vidually. Thus we are already imposing a structure onto semantics (that meanings are sets) that languages cannot override. All stands and falls with the question whether a language contains genuinely ambiguous expressions. A defender of the functional view will have to claim that expressions are not ambiguous in that sense; they simply mean what they mean in all their totality. This is difficult to maintain since it would deprive us of the possibility of differentiating between idiomatic and nonidiomatic meanings of expressions. The expression “He kicked the bucket” will have to have both the literal and the idiomatic reading as its meaning simpliciter without there being a way to say what it is that makes the idiomatic reading idiomatic.

Another problem with the functional account is that it assumes that all ambiguity is spurious. Suppose namely that there is a string  $\vec{x}$  that can be derived in several different ways. As the meaning of  $\vec{x}$  is assumed to be unique, we want each of the derivations to give us the unique reading. This is problematic for reasons of structural ambiguity.

(3.9)     \_\_ is square free or it is a product of two  
          prime numbers and greater than 100

This description can be read in two ways. It says that the number is greater than 100; or it is less or equal than 100 and then it is either square free of the product of two primes. Alternatively, the number is either square free or it is not, and in the latter case the product of two primes and greater than 100. In the second reading 71 satisfies the description, in the first reading it does not. The values for each of the readings can be obtained using a compositional grammar. However, the sum of all values cannot be so given, since it would require the grammar to know in each case about alternative readings. This cannot work. Of course, such a claim needs rigorous proof. We shall return to this matter in Section 3.5.

I also add another feature that is frequently encountered in artificial languages but not in human languages. I have above given an example of a language that figures in [Zadrozny, 1994] to show that there are languages which we intuitively consider not compositional. A critical analysis of this example reveals that the intuition is based on the assumption that what is graphically complex (here the string /ab/) also is syntactically complex. Since alphabets are small, ‘graphically complex’ cannot always mean: consists of more than one letter. Rather, it is taken to mean: consists of more than one identifier, where identifiers are sequences of letters not interrupted by special symbols. More complex criteria can be imag-

ined; what is important is that syntactic complexity is decidable regardless of the underlying grammar. That this is so is a *design property* of formal languages; it is built into the parser. We cannot likewise assume human languages to be built that way. The said property, that complexity is decidable on the basis of the string alone, is called **morphological transparency**. Human languages are therefore morphologically intransparent. Idioms are a case in point.

## 3.2 Interpreted Languages and Grammars

We assume the setup of the previous chapter. 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  $E$ .  $E$  can be, for example,  $A^*$ , but different choices are possible (and often necessary).

To differentiate languages as sets of strings from the interpreted languages defined below we shall call sets of strings **string languages** (though in fact we have allowed the exponents to be *sequences* of strings).

**Definition 3.1** *Let  $E$  and  $M$  be sets (of exponents and meanings, respectively). The members of  $E \times M$  are called **signs**. For a sign  $\sigma = \langle e, m \rangle$  define*

$$(3.10) \quad \varepsilon(\sigma) := e, \quad \mu(\sigma) := m$$

*$e$  is the **exponent** of  $\sigma$  and  $m$  its **meaning**. A set  $L \subseteq E \times M$  is called an **interpreted language over  $E$** . The projection*

$$(3.11) \quad \varepsilon[L] := \{e : \text{there is } m \in M : \langle e, m \rangle \in L\}$$

*is called the **string language of  $L$**  and the set*

$$(3.12) \quad \mu[L] := \{m : \text{there is } e \in E : \langle e, m \rangle \in L\}$$

*the **expressive power of  $L$** .*

The meaning of  $\sigma$  is not to be confused with its **denotation**, a term that I wish to avoid.

**Definition 3.2** Let  $L$  be an interpreted language.  $L$  is **unambiguous** if for every  $\langle e, m \rangle, \langle e, m' \rangle \in L$  we have  $m = m'$ .  $L$  is **monophone** if for every  $\langle e, m \rangle, \langle e, m' \rangle \in L$  we have  $e = e'$ .

Thus a language is generally defined to be a set of signs; that a sign is seen here just as a pair and not a triple (see Section 3.2) 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, 1967]. (Chomsky also endorsed that view in [Chomsky, 1993], though the exponents in Generative Grammar are far more complex.) 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 3.3** Let  $E$  be a set of exponents 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  $\mathcal{J}(f) : (E \times M)^{\Omega(f)} \hookrightarrow (E \times M)$ .

To put it somewhat more simply, given  $E$  and  $M$ , the set  $S := E \times M$  is the **space of signs**. If  $f$  is a function symbol,  $\mathcal{J}(f)$  is a partial  $n$ -ary function on  $S$ .

**Example 28.** (See also Example 5.) If  $G$  is a grammar,  $L(G)$  is either finite or countable. This is because we can effectively enumerate the terms, and there are only countably many terms. Let now  $L$  be countable. Then there is a bijection  $f : \mathbb{N} \rightarrow L$ . Define the grammar  $G$  in the same way as in Example 5. It is easy to see that the terms are of the form  $s^n b$  for some  $n \in \mathbb{N}$ . For this term we have  $\mathcal{J}(s^n b) = f(n)$ . Thus this grammar generates  $L$ . We conclude that a language has a grammar if and only if it is finite or countable.  $\odot$

We refer the reader to the Appendix A for the relationship between a partial function  $f : A \hookrightarrow B \times C$  and the projections  $\pi_B \circ f : A \hookrightarrow B$  and  $\pi_C \circ f : A \hookrightarrow C$ . We apply this to the case at hand. The symbol  $f$  is interpreted by a function  $\mathcal{J}(f) : (E \times M)^{\Omega(f)} \hookrightarrow (E \times M)$ , and so we can factor  $\mathcal{J}(f)$  into a *pair* of partial functions

$$(3.13) \quad f^e := \pi_E \circ \mathcal{J}(f), \quad f^m := \pi_M \circ \mathcal{J}(f)$$

This means in more detail that for all signs  $\sigma_i, i < \Omega(f)$ , we put

$$(3.14) \quad \begin{aligned} f^\varepsilon(\sigma_0, \dots, \sigma_{\Omega(f)-1}) &:= \varepsilon(\mathcal{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1})) \\ f^\mu(\sigma_0, \dots, \sigma_{\Omega(f)-1}) &:= \mu(\mathcal{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1})) \end{aligned}$$

It follows that we have

$$(3.15) \quad \mathcal{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1}) = \langle f^\varepsilon(\sigma_0, \dots, \sigma_{\Omega(f)-1}), f^\mu(\sigma_0, \dots, \sigma_{\Omega(f)-1}) \rangle$$

This is written in a more concise form as

$$(3.16) \quad \mathcal{J}(f) = f^\varepsilon \star f^\mu$$

Here,  $f \star g$  for  $f : A^n \rightarrow C$  and  $g : B^n \rightarrow D$  is a function from  $(A \times B)^n$  to  $C \times D$  defined by

$$(3.17) \quad (f \star g)(\langle x_0, y_0 \rangle, \dots, \langle x_{n-1}, y_{n-1} \rangle) := \langle f(x_0, \dots, x_{n-1}), g(y_0, \dots, y_{n-1}) \rangle$$

(Notice that we write  $f(x_0, \dots, x_{n-1})$  in place of  $f(\langle x_0, \dots, x_{n-1} \rangle)$ .) Now, in place of a single interpretation function  $\mathcal{J}$  we may also consider having *two* such functions, namely  $\mathcal{J}^\varepsilon$  and  $\mathcal{J}^\mu$ , which we get as follows.

$$(3.18) \quad \mathcal{J}^\varepsilon(f) := \pi_E \circ \mathcal{J}(f), \quad \mathcal{J}^\mu(f) := \pi_M \circ \mathcal{J}(f)$$

As we shall see, having two independent interpretations changes things dramatically. So we shall give the new construct a name and call in *bigrammar*.

**Definition 3.4** *Let  $E$  be a set of exponents and  $M$  a set of meanings. A **bigrammar** over  $E$  and  $M$  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 concept of a bigrammar is a different concept as we shall show. If  $G = \langle \Omega, \mathcal{J}^\varepsilon, \mathcal{J}^\mu \rangle$  is a bigrammar then put  $\mathcal{J}(f) := \mathcal{J}^\varepsilon(f) \times \mathcal{J}^\mu(f)$ . Then  $G_\times := \langle \Omega, \mathcal{J} \rangle$  is an interpreted grammar. Conversely, given an interpreted grammar  $G = \langle \Omega, \mathcal{J} \rangle$ , put  $G^\times := \langle \Omega, \mathcal{J}^\varepsilon, \mathcal{J}^\mu \rangle$ ; this is a bigrammar.

It is easy to see that for every interpreted grammar  $G$ ,  $G = (G^\times)_\times$ . However, it is not generally the case that  $H = (H_\times)^\times$  for a bigrammar. This is because an



interpreted grammar  $G = \langle \Omega, \mathcal{J} \rangle$  can be turned into a bigrammar in several ways. Notice namely that

$$(3.19) \quad \text{dom}(\mathcal{J}^\varepsilon(f) \star \mathcal{J}^\mu(f)) = \text{dom}(\mathcal{J}^\varepsilon(f)) \cap \text{dom}(\mathcal{J}^\mu(f))$$

However, the grammar  $G_\times$  has the property that

$$(3.20) \quad \text{dom}(f^\varepsilon) = \text{dom}(f^\mu)$$

Hence, a bigrammar of the form  $G^\times$  satisfies

$$(3.21) \quad \text{dom}(\mathcal{J}^\varepsilon(f)) = \text{dom}(\mathcal{J}^\mu(f))$$

We call a bigrammar satisfying (3.21) **balanced**.

The terminology of Section 2.1 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 3.3). 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:

$$(3.22) \quad \iota_G(f s_0 s_1 \cdots s_{\Omega(f)-1}) := \mathcal{J}(f)(\iota_G(s_0), \iota_G(s_1), \cdots, \iota_G(s_{\Omega(f)-1}))$$

We use also the following notation. For terms  $t$  we let  $t^\varepsilon$  be the exponent of  $\iota(t)$  and  $t^\mu$  the meaning. A term is **semantically definite** if  $t^\mu$  exists; and **orthographically definite** if  $t^\varepsilon$  exists. We say that  $t$  is **definite** if it is both orthographically and semantically definite and **indefinite** otherwise. In a balanced bigrammar a term is definite iff it is semantically definite iff it is orthographically definite. In general however they are different, but only slightly. For a term of the form  $t = f(u_0, \cdots, u_{\Omega(f)-1})$  we either have that one of the  $u_i$  is not definite, in which case  $t$  is both semantically and orthographically indefinite. Or all of the  $u_i$  are definite, and then  $t$  can be orthographically but not semantically definite, or semantically but not orthographically definite.

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  $\langle \sigma_0, \sigma_1, \cdots \rangle$  of signs  $\iota(t)$  computes the value of  $t$  where for every  $i \in \mathbb{N}$ ,  $x_i$  is interpreted as  $\sigma_i$ .

Figure 3.1: A Grammar for Binary Strings

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

**Example 29.**  $A := \{\mathbf{0}, \mathbf{1}, +, -, (, ), =\}$ .  $F := \{f_0, f_1, f_2, f_3, f_4, f_5, f_6\}$ .  $\Omega(f_0) := \Omega(f_1) := 0$ ,  $\Omega(f_2) := \Omega(f_3) := 1$ ,  $\Omega(f_4) := \Omega(f_5) := \Omega(f_6) := 2$ .  $\vec{x}$  is **binary** if it only contains  $/\mathbf{0}/$  and  $/\mathbf{1}/$ ;  $\vec{x}$  is a **term** if it does not contain  $/=/$ . The grammar is shown in Figure 3.1. 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  $\perp$ ).  $\otimes$

**Example 30.** We shall now define an unbalanced bigrammar that defines the same interpreted grammar as the previous example. The semantic functions are shown in Figure 3.3. For the bigrammar  $G = \langle \Omega, \mathcal{K}^\varepsilon, \mathcal{K}^\mu \rangle$  we find that  $G^\times = \langle \Omega, \mathcal{J} \rangle$ . However, it does not satisfy the equations (3.21). For example,  $\mathcal{K}^\varepsilon(f_2)(\langle (1+1), 2 \rangle)$  is undefined while  $\mathcal{K}^\mu(f_2)(\langle (1+1), 2 \rangle) = 4$ , since  $\mathcal{K}^\mu$  does not look at the exponent.

Figure 3.2: An Unbalanced Bigrammar for Binary Strings I

$$\begin{aligned}
\mathcal{K}^\varepsilon(f_0)() &:= \mathbf{0} \\
\mathcal{K}^\varepsilon(f_1)() &:= 1 \\
\mathcal{K}^\varepsilon(f_2)(\langle \vec{x}, n \rangle) &:= \begin{cases} \vec{x}\mathbf{0} & \text{if } \vec{x} \text{ is binary} \\ \text{undefined} & \text{else} \end{cases} \\
\mathcal{K}^\varepsilon(f_3)(\langle \vec{x}, n \rangle) &:= \begin{cases} \vec{x}\mathbf{1} & \text{if } \vec{x} \text{ is binary} \\ \text{undefined} & \text{else} \end{cases} \\
\mathcal{K}^\varepsilon(f_4)(\langle \vec{x}, n \rangle, \langle \vec{y}, m \rangle) &:= \begin{cases} (\vec{x}+\vec{y}) & \text{if } \vec{x}, \vec{y} \text{ are terms} \\ \text{undefined} & \text{else} \end{cases} \\
\mathcal{K}^\varepsilon(f_5)(\langle \vec{x}, n \rangle, \langle \vec{y}, m \rangle) &:= \begin{cases} (\vec{x}-\vec{y}) & \text{if } \vec{x}, \vec{y} \text{ are terms} \\ \text{undefined} & \text{else} \end{cases} \\
\mathcal{K}^\varepsilon(f_6)(\langle \vec{x}, n \rangle, \langle \vec{y}, m \rangle) &:= \begin{cases} \vec{x}=\vec{y} & \text{if } \vec{x}, \vec{y} \text{ are terms} \\ \text{undefined} & \text{else} \end{cases}
\end{aligned}
\tag{3.24}$$

Notice that the semantic functions are not total, but could easily be made to be. Notice also that they do not depend on the exponent, so they can be further simplified. This will be discussed in detail in Section 3.3.  $\odot$

Let me conclude with a few words on the algebraic treatment. A grammar  $G = \langle \Omega, \mathcal{J} \rangle$  can also be viewed as a partial  $\Omega$ -algebra defined over the space  $E \times M$  (see Appendix A for definitions). Bigrammars have no straightforward algebraic equivalent. Exercises 33 and 34 will pursue this theme.

**Exercise 25.** It is possible to interpret the modes  $f_2$  and  $f_3$  by the string functions  $\vec{x} \mapsto \mathbf{0}\vec{x}$  and  $\vec{x} \mapsto \mathbf{1}\vec{x}$ . Show that it is however impossible to use the meaning functions as above with these string functions.

**Exercise 26.** (Continuing the previous exercise.) Give a grammar that generates the language of equations using the string functions above. (Evidently, the

Figure 3.3: An Unbalanced Bigrammar for Binary Strings II

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

functions on meanings must be quite different.)

**Exercise 27.** Let  $G = \langle \Omega, \mathcal{J} \rangle$  be a grammar. Show that there is a bigrammar  $G_\bullet = \langle \Omega, \mathcal{J}_\bullet^\varepsilon, \mathcal{J}_\bullet^\mu \rangle$  such that  $(G_\bullet)^\times = G$  and for every  $f \in F$ ,  $\mathcal{J}_\bullet^\varepsilon(f)$  is total. (Dually, we can construct  $G_\bullet$  such that  $\mathcal{J}_\bullet^\mu(f)$  is total for every  $f \in F$ .)

**Exercise 28.** (Using the previous exercise.) Show by giving an example that we cannot expect both  $\mathcal{J}_\bullet^\varepsilon(f)$  and  $\mathcal{J}_\bullet^\mu(f)$  to be total. *Hint.* This should be totally straightforward.

### 3.3 Compositionality and Independence

In this section we shall look at the interdependence between the components of a sign. We shall look at ways of formulating the grammar in such a way that the exponents and meanings are completely independent. We have so far assumed that the modes are interpreted as functions on signs. As such they have the form

$$(3.26) \quad \mathcal{J}(f) = f^\varepsilon \star f^\mu$$

with the functions defined as given in (3.21). If, however, we start with a bigrammar we simply put

$$(3.27) \quad f^\varepsilon := \mathcal{J}^\varepsilon(f), \quad f^\mu := \mathcal{J}^\mu(f)$$

In this case, as we observed, (3.21) does not necessarily hold any more. Although we do not always mention this fact, the reader is asked to be aware of the possibility of using a bigrammar in place of a grammar, which may open more possibilities to define grammars.

There are two senses in which these equations can be required to hold. I call the first the *strict sense*. The equations are valid as stated above. That means that the equations specified are valid even if the relevant functions are defined on signs that may not be in the language. The *extensional sense* requires that the equations only hold for the language of the grammar. This is expressed as

$$(3.28) \quad \mathcal{J}(f) \upharpoonright L(G) = (f^\varepsilon \star f^\mu) \upharpoonright L(G)$$

where if  $f : A^n \rightarrow B$  and  $C \subseteq A$ ,

$$(3.29) \quad f \upharpoonright C := \{\langle \vec{c}, f(\vec{c}) \rangle : \vec{c} \in C\}$$

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 new words into a language or create new idioms. 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. Consider for example the nouns of Malay (cf. also the discussion in Example 34 below). Malay nouns reduplicate in the plural. Now suppose a new word, say, a loanword from English

is introduced. Will it be reduplicated or will it be used with the English plural? Exactly such question is raised in the so-called “wug-test”, where people are asked to form the plural of a word that is not English (and is not known to be a word of any other language). If speaker A forms a plural it means that his morphological functions are more general; they operate on word that are not English, and they operate even in the absence of any semantics that goes with this word. Children are in a similar situation. When they grow up they will have to guess how the plural of nouns is formed. It is not realistic to assume that they will simply learn the plural of each word individually. Rather, they will abstract a general rule that can be used on new words as well. And they can both understand what is a morphological plural and what is the concept behind plurality. And both seem to be independent. Notice that the idea of a human grammar as different from a formal grammar is irrelevant here. Formal languages often do display similar differences. And though the wug-test seems to indicate that there is a uniform rule of plural formation in English it is not clear that all people have the same abstract formation rule. Not only does individual variation exist (showing us extensional differences, that is, differences in the languages of the speakers); also it is quite conceivable that intensional variation exists. In other words, it is conceivable that when presented with a nonexistent verbal root, German speakers will differ as to how it will be inflected even when they otherwise agree on existing verbs (though I am not aware of a positive result showing this).

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  $f$ ,  $\mathcal{J}'(f) \upharpoonright L(G) = \mathcal{J}(f) \upharpoonright L(G)$ . 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 (3.26) above. I shall rewrite it as follows:

$$(3.30) \quad \begin{aligned} & \mathcal{J}(f)(\langle e_0, m_0 \rangle, \dots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1} \rangle) \\ &= \langle f^e(\langle e_0, m_0 \rangle, \dots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1} \rangle), \\ & \quad f^\mu(\langle e_0, m_0 \rangle, \dots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1} \rangle) \rangle \end{aligned}$$

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

be restated as follows. (For notions of independence, see the Appendix A. For partial functions, independence is weak independence by default.)

**Definition 3.5** A bigrammar  $G$  is **semicompositional** if for every mode  $f$ ,  $f^\mu$  is independent of the exponents of the signs. If the  $f^\mu$  are strongly independent of the exponents,  $G$  is called **compositional**.  $G$  is **extensionally compositional** if it has an extensional variant that is compositional. An interpreted language  $L$  is **compositional** if there is a compositional bigrammar  $G$  such that  $L = L(G)$ .

We extend these notions to interpreted grammars as follows. For an interpreted grammar  $G$ ,  $G$  is  $\mathcal{P}$  if and only if  $G_\times$  is  $\mathcal{P}$  (see Page 88 for notation). So,  $G$  is semicompositional if and only if  $G_\times$  is. Notice that a language is compositional if and only if it has a compositional interpreted grammar.

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

$$(3.31) \quad \mu(\mathcal{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1})) \stackrel{\geq}{=} f_*^\mu(\mu(\sigma_0), \dots, \mu(\sigma_{\Omega(f)-1}))$$

The sign  $\stackrel{\geq}{=}$  means that the left and right hand sides are equal *if defined*; and moreover, the right side is defined if the left hand side is, but the converse need not hold. If  $G$  is compositional then also the left hand side is defined if the right hand side is, so full equality holds. In that case we can put

$$(3.32) \quad f_*^\mu(m_0, \dots, m_{\Omega(f)-1}) := f^\mu(\langle e, m_0 \rangle, \langle e, m_1 \rangle, \dots, \langle e, m_{\Omega(f)-1} \rangle)$$

where  $e$  is chosen arbitrarily. Since by assumption  $f^\mu$  does not depend on the exponents, any choice of  $e$  will give the same result. 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

$$(3.33) \quad \mu[g] := \{\langle \mu(\sigma_0), \dots, \mu(\sigma_n) \rangle : \langle \sigma_0, \dots, \sigma_n \rangle \in g\}$$

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

$$(3.34) \quad f_*^\mu := \mu[\mathcal{J}(f)]$$

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  $t = fs_0 \cdots s_{\Omega(f)-1}$  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

$$(3.35) \quad \widehat{f^\varepsilon} := (\varepsilon \circ \mathcal{J}(f)) \upharpoonright L(G)$$

This function is defined exactly on the signs of  $L(G)$ . Now take as  $\widehat{f_*^\mu}$  any function extending  $f_*^\mu$ .

**Example 31.** Here is an example. Let  $G = \langle \Omega, \mathcal{J} \rangle$  be a grammar containing a binary mode  $f$ , and a unary modes  $g_i$ ,  $i < 3$ , where

$$(3.36) \quad \begin{aligned} \mathcal{J}(g_0)(\circ) &= \langle \text{ed}, \text{past}' \rangle \\ \mathcal{J}(g_1)(\circ) &= \langle \text{laugh}, \text{laugh}' \rangle \\ \mathcal{J}(g_2)(\circ) &= \langle \text{car}, \text{car}' \rangle \end{aligned}$$

$$(3.37) \quad \mathcal{J}(f)(\langle e, m \rangle, \langle e', m' \rangle) := \langle e \frown e', m'(m) \rangle$$

We now extend  $f^\mu$  in such a way that it also takes the pair  $\langle \text{car}', \text{past}' \rangle$  and returns some value. Then put

$$(3.38) \quad \widehat{f^\varepsilon}(e, e') := \begin{cases} e \frown e' & \text{unless } e = \text{car} \text{ and } e' = \text{ed} \\ \text{undefined} & \end{cases}$$

This grammar generates the same output language. ⊛

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 notion 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.

Signs are pairs; switching the order in the pair gives rise to the **dual** of the sign. Switching the order in the entire language defines the dual of the language. Notice that most technical notions do not distinguish between exponents and meanings, so they can be applied to both a language and its dual. The notion dual to compositionality is known as *autonomy*.

**Definition 3.6** A bigrammar  $G$  is **semiautonomous** if for every mode  $f$  the function  $f^\varepsilon$  is weakly independent of the  $m_i$ . If  $f^\varepsilon$  are also strongly independent of them  $m_i$ ,  $G$  is called **autonomous**.  $G$  is **extensionally autonomous** if it has an extensional variant that is autonomous. An interpreted language  $L$  is **autonomous** if there is an autonomous bigrammar  $G$  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

$$(3.39) \quad \varepsilon(\mathcal{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1})) \doteq f_*^\varepsilon(\varepsilon(\sigma_0), \dots, \varepsilon(\sigma_{\Omega(f)-1}))$$

Again, the left and side is 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 3.7** A bigrammar is **independent** if it is both compositional and autonomous; it is **extensionally independent** if it is both extensionally compositional and extensionally autonomous. A language is **independent** if it has an independent bigrammar.

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

$$(3.40) \quad \mathcal{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1}) = \langle f_*^\varepsilon(e_0, \dots, e_{\Omega(f)-1}), f_*^\mu(m_0, \dots, m_{\Omega(f)-1}) \rangle$$

with the left hand side defined if and only if the right hand side is. Another formulation is as follows:

$$(3.41) \quad \mathcal{J}(f) = (f_*^\varepsilon \circ \overbrace{\langle \varepsilon, \dots, \varepsilon \rangle}^{\Omega(f)}) \times (f_*^\mu \circ \overbrace{\langle \mu, \dots, \mu \rangle}^{\Omega(f)})$$

or

$$(3.42) \quad \mathcal{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1}) \\ = \langle f_*^\varepsilon(\varepsilon(\sigma_0), \dots, \varepsilon(\sigma_{\Omega(f)-1})), f_*^\mu(\mu(\sigma_0), \dots, \mu(\sigma_{\Omega(f)-1})) \rangle$$

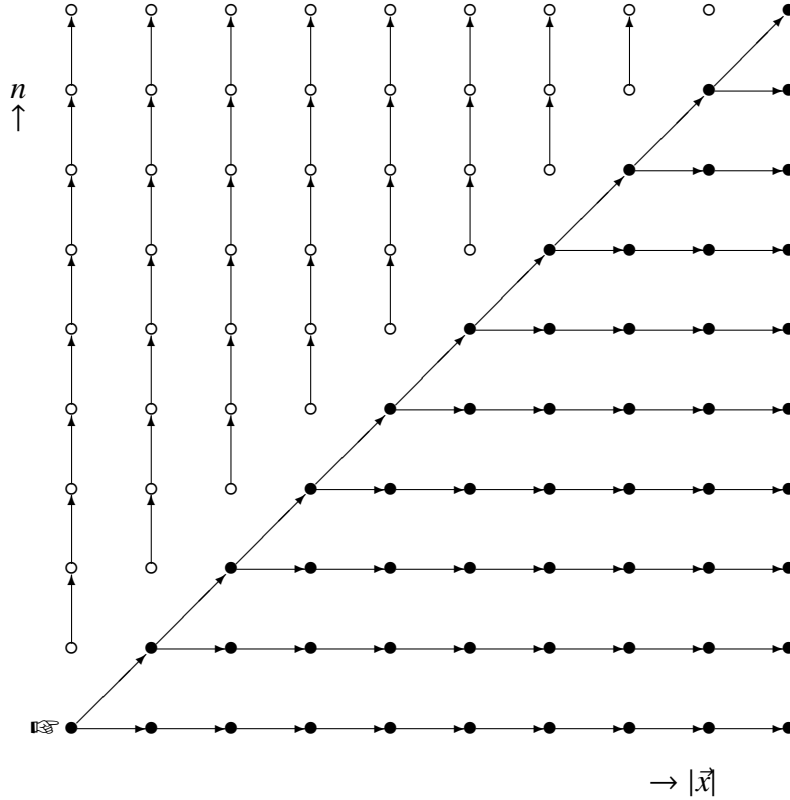
It may be thought that extensional independence follows from extensional autonomy and extensional compositionality. However, this is not so.

**Example 32.** We construct four different grammars to show that autonomy and compositionality are independent notions. Let  $A := \{\mathbf{a}\}$ ,  $E := A^*$ ;  $M := \mathbb{N}$ . The signature is  $\{f_0, f_1, f_2\}$ , with  $f_0$  nullary and  $f_1$  and  $f_2$  both unary. We have

$$(3.43) \quad \begin{aligned} \mathcal{J}(f_0)() &:= \langle \varepsilon, 0 \rangle \\ \mathcal{J}(f_1)(\langle \vec{x}, n \rangle) &:= \begin{cases} \langle \vec{x} \hat{\ } \mathbf{a}, n + 1 \rangle & \text{if } |\vec{x}| = n \\ \text{undefined} & \text{otherwise} \end{cases} \\ \mathcal{J}(f_2)(\langle \vec{x}, n \rangle) &:= \begin{cases} \langle \vec{x} \hat{\ } \mathbf{a}, n \rangle & \text{if } |\vec{x}| \geq n \\ \langle \vec{x}, n + 1 \rangle & \text{otherwise} \end{cases} \end{aligned}$$

Call this grammar  $U$ . The action of the unary functions on the space  $E \times M$  is shown in Figure 3.4.  $U$  generates the language  $D := \{\langle \vec{x}, n \rangle : n \leq |\vec{x}|\}$ , as is easily verified; the entry point is the origin, and everything is in  $D$  that is reachable by following the arrows. Notice that the second clause of the definition for  $\mathcal{J}(f_2)$  is never used inside  $D$ . Thus, we could have made  $\mathcal{J}(f_2)(\langle \vec{x}, n \rangle)$  undefined if  $n > |\vec{x}|$ . That would give us an extensional variant of the original grammar.  $U$  is not autonomous:  $\mathcal{J}(f_2)(\langle \mathbf{a}, 3 \rangle) = \langle \mathbf{a}, 4 \rangle$ , but  $\mathcal{J}(f_2)(\langle \mathbf{a}, 1 \rangle) = \langle \mathbf{a}\mathbf{a}, 1 \rangle$ . So to compute the exponent we need to know the meaning. It is not compositional either. For we

Figure 3.4: The Action of the Grammar  $U$



have  $\mathcal{J}(f_2)(\langle \text{aaa}, 3 \rangle) = \langle \text{aaaa}, 3 \rangle$ , 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  $\mathcal{J}$ :

$$(3.44) \quad \begin{aligned} \mathcal{J}^a(f_2)(\langle \vec{x}, n \rangle) &:= \begin{cases} \langle \vec{x} \hat{a}, n \rangle & \text{if } |\vec{x}| \geq n \\ \langle \vec{x} \hat{a}, n + 1 \rangle & \text{else} \end{cases} \\ \mathcal{J}^c(f_2)(\langle \vec{x}, n \rangle) &:= \begin{cases} \langle \vec{x} \hat{a}, n \rangle & \text{if } |\vec{x}| \geq n \\ \langle \vec{x} \hat{a} \hat{a}, n \rangle & \text{else} \end{cases} \\ \mathcal{J}^{ac}(f_2)(\langle \vec{x}, n \rangle) &:= \langle \vec{x} \hat{a}, n \rangle \end{aligned}$$

All of them only generate the language  $D$ . The grammar  $U^{ac} := \langle \Omega, \mathcal{J}^{ac} \rangle$  is au-

tonomous and compositional (even independent).  $U^c = \langle \Omega, \mathcal{J}^c \rangle$  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  $\varepsilon(\mathcal{J}^c(f_2)(\langle aa, 2 \rangle)) = aaa \neq aa = \varepsilon(\mathcal{J}^c(f_2)(\langle aa, 3 \rangle))$ . Similarly we find that  $\langle U^a := \langle \Omega, \mathcal{J}^a \rangle$  is autonomous but not compositional.  $\odot$

Finally, let us return to bigrammars. If a bigrammar 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

$$(3.45) \quad \mathcal{J}_\varepsilon(f)(\vec{\sigma}) = g(e_0, \dots, e_{\Omega(f)-1})$$

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

$$(3.46) \quad \begin{aligned} \mathcal{J}_\varepsilon^*(f)(\vec{\sigma}) &:= g^*(e_0, \dots, e_{\Omega(f)-1}) \\ \mathcal{J}_\mu^*(f) &:= \mathcal{J}_\mu(f) \cap \text{dom}(\mathcal{J}_\varepsilon(f)) \end{aligned}$$

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

$$(3.47) \quad \begin{aligned} \langle \mathcal{J}_\varepsilon^*(f)(\vec{\sigma}), \mathcal{J}_\mu^*(f)(\vec{\sigma}) \rangle &= \langle g^*(\vec{\sigma}), \mathcal{J}_\mu(\vec{\sigma}) \rangle \\ &= \langle g(\vec{\sigma}), \mathcal{J}_\mu(\vec{\sigma}) \rangle \\ &= \langle \mathcal{J}_\varepsilon(\vec{\sigma}), \mathcal{J}_\mu(\vec{\sigma}) \rangle \end{aligned}$$

**Example 33.** From a grammar we can essentially make two bigrammars: 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  $A := \{a\}$ ,  $E := A^*$ ;  $M := \mathbb{N}$ . The signature is  $\{f_0, f_1, f_2\}$ , with  $f_0$  nullary and  $f_1$  and  $f_2$  both unary.

$$(3.48) \quad \begin{aligned} \mathcal{J}(f_0)() &:= \langle \varepsilon, 0 \rangle \\ \mathcal{J}(f_1)(\langle \vec{x}, n \rangle) &:= \langle \vec{x}^{\sim} a, n + 1 \rangle \\ \mathcal{J}(f_2)(\langle \vec{x}, n \rangle) &:= \begin{cases} \langle \vec{x}^{\sim} a, n \rangle & \text{if } |\vec{x}| = n \\ \text{undefined} & \text{else} \end{cases} \end{aligned}$$

The definite terms are of the form  $f_1^n f_0$  or  $f_2 f_1^n f_0$ . The first bigrammar is as follows.

$$(3.49) \quad \begin{aligned} \mathcal{J}_\varepsilon^*(f_1)(\langle \vec{x}, n \rangle) &:= \vec{x}^{\sim} a \\ \mathcal{J}_\varepsilon^*(f_2)(\langle \vec{x}, n \rangle) &:= \begin{cases} \vec{x}^{\sim} a & \text{if } |\vec{x}| = n \\ \text{undefined} & \text{else} \end{cases} \end{aligned}$$

$$(3.50) \quad \begin{aligned} \mathcal{J}_\mu^\times(f_1)(\langle \vec{x}, n \rangle) &:= n + 1 \\ \mathcal{J}_\mu^\times(f_2)(\langle \vec{x}, n \rangle) &:= n \end{aligned}$$

The second bigrammar is as follows.

$$(3.51) \quad \begin{aligned} \mathcal{J}_\varepsilon^\times(f_1)(\langle \vec{x}, n \rangle) &:= \vec{x} \hat{\ } a \\ \mathcal{J}_\varepsilon^\times(f_2)(\langle \vec{x}, n \rangle) &:= \vec{x} \hat{\ } a \end{aligned}$$

$$(3.52) \quad \begin{aligned} \mathcal{J}_\mu^\times(f_1)(\langle \vec{x}, n \rangle) &:= n + 1 \\ \mathcal{J}_\mu^\times(f_2)(\langle \vec{x}, n \rangle) &:= \begin{cases} n & \text{if } |\vec{x}| = n \\ \text{undefined} & \text{else} \end{cases} \end{aligned}$$

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

$$(3.53) \quad \mathcal{J}_\varepsilon^\times(f_2)(\langle \text{aaa}, 2 \rangle) = \text{undefined}, \quad \mathcal{J}_\varepsilon^\times(f_2)(\langle \text{aaa}, 3 \rangle) = \text{aaaa}$$

Thus  $G^\times$  is indeed semiautonomous but compositional. Likewise for the other claim. However, it turns out that there is no bigrammar corresponding to  $G$  that is both autonomous and compositional. To see this, suppose  $G^{\times\text{d}} = \langle \Omega, \mathcal{J}_\varepsilon^{\times\text{d}}, \mathcal{J}_\mu^{\times\text{d}} \rangle$  is such a grammar. Then for any given string  $\vec{x}$  there is some  $n$  (namely  $|\vec{x}|$ ) such that  $\mathcal{J}_\varepsilon^{\times\text{d}}(f_2)(\langle \vec{x}, n \rangle)$  is defined. If the grammar is autonomous this means that for all  $m$   $\mathcal{J}_\varepsilon^{\times\text{d}}(f_2)(\langle \vec{x}, m \rangle)$  is defined. Hence the function  $\mathcal{J}_\varepsilon^{\times\text{d}}(f_2)$  is total. Likewise we see that  $\mathcal{J}_\mu^{\times\text{d}}(f_2)$  is total. It follows that  $\text{dom}(\mathcal{J}^{\times\text{d}}(f_2)) = \text{dom}(\mathcal{J}(f_2))$  is total. But this is not the case in  $G$ .  $\text{⊛}$

The independence of form and meaning has interesting consequences also for the assessment of arguments concerning generative capacity. Both examples concern the problem whether or not there is copying in syntax.

**Example 34.** This and the next example deal with the problem of reduplication. In Malay, the plural of a noun is formed by reduplicating it: /orang/ means ‘man’,

/orang-orang/ means ‘men’ (see also the discussion on Page 73). Thus, the plural mode  $p$  in Malay is unary mode and is interpreted as follows.

$$(3.54) \quad \mathcal{J}(p)(\langle e, m \rangle) := \begin{cases} \langle e^{\wedge} - \wedge e, \text{pl}(m) \rangle & \text{if } e \text{ is a noun} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Under this interpretation, there is a plural morpheme with no fixed exponent; the exponent of the morpheme depends on whatever the singular is. If Malay works like this, then the grammar is not context free in the sense that it has non context free rules. An alternative view however is to assume that Malay has a binary operation  $q$  with the following interpretation.

$$(3.55) \quad \mathcal{J}(q)(\langle e, m \rangle, \langle e', m' \rangle) := \begin{cases} \langle e^{\wedge} - \wedge e', \text{pl}(m) \rangle & \text{if } e \text{ and } e' \text{ are nouns} \\ & \text{and } e = e' \\ \text{undefined} & \text{otherwise} \end{cases}$$

This means that each occurrence of the singular form is a true occurrence of a constituent. A third account is this. Malay has a binary mode  $r$  defined by

$$(3.56) \quad \mathcal{J}(r)(\langle e, m \rangle, \langle e', m' \rangle) := \begin{cases} \langle e^{\wedge} - \wedge e', \text{pl}(m) \rangle & \text{if } e \text{ and } e' \text{ are nouns} \\ & \text{and } m = m' \\ \text{undefined} & \text{otherwise} \end{cases}$$

This looks similar to  $q$  but the difference is that the combinatorial restrictions are now semantic rather than syntactic. This has repercussions on how powerful we believe the syntax of Malay is. If we think Malay uses  $p$  then the syntax uses nonlinear polynomials, hence cannot be approximated by what is known as linear context free rewrite systems (LCFRS). If we think that Malay uses  $q$  we think that the syntax is an LCFRS, even context free, since the number of nouns is finite. However, performing the substitution tests will reveal that there are as many form classes as there are nouns. Finally, if we think that Malay uses  $r$  we think that the syntax is context free *and* that there is essentially only one noun class. It is not easy to distinguish between these alternatives. Only if Malay has two nouns  $e$  and  $e'$  with identical meaning can we check whether Malay uses  $p$  or  $q$  (though it is in principle also possible to treat exceptions with extra modes as well).  $\otimes$

**Example 35.** [Manaster-Ramer, 1986] discusses a construction of English in which a constituent is repeated verbatim:

(3.57) The North Koreans were developing nuclear weapons  
anyway, Iraq war or no Iraq war.

(3.58) \*The North Koreans were developing nuclear weapons  
anyway, Iraq war or no Afghanistan war.

The meaning is something like: “independent of”, “irrespective of”. As Manaster-Ramer claims, the construction has the form ( $\vec{x}$  or no  $\vec{x}$ ), where  $\vec{x}$  is an NP (determinerless!). The construction / $\vec{x}$  or no  $\vec{y}$ / where  $\vec{x}$  and  $\vec{y}$  are different does not have this meaning. On this basis, Manaster-Ramer argues that English is not context free. Basically, the idea is that there is a unary mode  $f$  defined as follows.

$$(3.59) \quad \mathcal{J}(f)(\langle e, m \rangle) := \begin{cases} \langle e \_ \text{or} \_ \text{no} \_ e, \text{irrespective of}(m) \rangle & \text{if } e \text{ is an NP} \\ \text{undefined} & \text{otherwise} \end{cases}$$

I put aside the alternative with a binary operation that checks for string identity. This is called the ‘X-or-no-X construction’ by [Pullum and Rawlins, 2007]. They observe that the second part of it need not be an exact copy. They take this as evidence that this is not a requirement imposed by the syntax but a semantic requirement. So the construction takes the form ‘X or no Y’, where X and Y may be different but must be synonymous. I shall leave this point aside. What [Pullum and Rawlins, 2007] propose is that rather than checking syntactic identity, English works with a binary mode  $g$  defined by

$$(3.60) \quad \mathcal{J}(f)(\langle e, m \rangle, \langle e', m' \rangle) := \begin{cases} \langle e \_ \text{or} \_ \text{no} \_ e, & \text{if } e, e' \text{ are NP} \\ \text{irrespective of}(m) \rangle & \text{and } m = m' \\ \text{undefined} & \text{otherwise} \end{cases}$$

The problem is reminiscent of reduplication discussed earlier. Although [Pullum and Rawlins, 2007] show that the resulting language is not context free, their argument makes clear that there are two notions of generative capacity involved. One is the purely syntactic capacity and the other is the capacity to generate signs. Given a bigrammar  $\langle \Omega, \mathcal{J}^e, \mathcal{J}^\mu \rangle$  we may either look at the language generated by  $\langle \Omega, \mathcal{J}^e \rangle$  (pure syntax), or we may look at the language  $\varepsilon[L(G)]$ . The first is the set of all syntactically well-formed sentences, the second the set of all syntactically and semantically well-formed sentences.

The two analyses are not identical. Suppose namely we have expressions that are synonymous for all we know (say /Abelian group/ and /commutative group/ then the two proposals make different claims about grammaticality. If syntactic identity is the key then using the expression

(3.61) Abelian group or no commutative group

cannot mean “irrespective of an abelian group”, whereas if semantic identity counted, that would be perfect. I have not investigated this, though.  $\otimes$

Under the assumption of independence it is possible to extend some of the results of formal language theory to the present setting. I give an instructive example. A CFL has the following property:

**Lemma 3.8 (Pumping Lemma)** *Let  $L$  be CF. Then there exists a number  $c_L$  such that for every  $\vec{x} \in L$  of length at least  $c_L$  there are strings  $\vec{u}, \vec{v}, \vec{w}, \vec{y}, \vec{z}$  such that*

1.  $\vec{x} = \vec{u}\vec{y}\vec{v}\vec{z}\vec{w}$ ;
2.  $\vec{x}\vec{y} \neq \varepsilon$ ;
3. for all  $n \in \mathbb{N}$ :  $\vec{u}\vec{y}^n\vec{v}\vec{z}^n\vec{w} \in L$ .

For a proof see among other [Harrison, 1978]. This theorem has many strengthenings, and all of them could be used in its place below. To be able to state the extension properly, we need to look at two different equivalence relations induced by a bigrammar  $\langle \Omega, \mathcal{J}^\varepsilon, \mathcal{J}^\mu \rangle$ . Recall from Definition 2.14 the definition of a categorical equivalence. The first is the equivalence  $\sim_{G^\varepsilon}$ , where  $G^\varepsilon := \langle G, \mathcal{J}^\varepsilon \times 1 \rangle$ , where  $1(f)$  gives a unit value for every input (and is always defined). This equivalence relation gives rise to the syntactic categories only. Another is the equivalence  $\sim_G$ , induced by  $G$  itself. It is defined in the same way as Definition 2.14, the only difference being that the definition is applied to a bigrammar. We say that  $G$  is **syntactically well regimented** if  $\sim_G = \sim_{G^\varepsilon}$ . Intuitively, if a grammar is syntactically well regimented then the combinability of signs can be determined by looking at the exponents alone (which does *not* mean that the semantic functions have to be total). Or,  $\mathcal{J}(f)(\vec{\sigma})$  is defined if only  $\mathcal{J}^\varepsilon(\vec{\sigma})$  is defined.

**Theorem 3.9** *Let  $L$  be an interpreted language that has a syntactically well regimented CF bigrammar. Then there is a  $c_L$  such that for all  $\langle \vec{x}, m \rangle \in L$  where  $\vec{x}$  has*



length at least  $c_L$  there are strings  $\vec{u}, \vec{v}, \vec{w}, \vec{y}, \vec{z}$ , an element  $n \in M$  and unary partial functions  $f, g$  on  $M$  such that

1.  $\langle \vec{x}, m \rangle = \langle \vec{u}\vec{y}\vec{v}\vec{z}\vec{w}, f(p) \rangle$ ;
2.  $\vec{x}\vec{y} \neq \varepsilon$ ;
3. for all  $n \in \mathbb{N}$ :  $\langle \vec{u}\vec{y}^n\vec{v}\vec{z}^n\vec{w}, f(g^n(p)) \rangle \in L$ .

The proof of the theorem proceeds basically in the same way as the proof of the original Pumping Lemma. Given a string  $\vec{x}$  we find a decomposition of the string; furthermore, we know that the decomposition is in terms of constituents. In other words, we have terms  $r(x_0)$ ,  $s(x_0)$  and a constant term  $t$  such that

1.  $\vec{x} = r^\varepsilon(s^\varepsilon(t^\varepsilon))$
2.  $\vec{y}\vec{v}\vec{z} = s^\varepsilon(t^\varepsilon)$
3.  $\vec{v} = t^\varepsilon$ .

Put  $p := t^\mu$ ,  $g(x_0) := s^\mu(x_0)$ , and  $f(x_0) := r^\mu(x_0)$ . This defines the functions. The assumption of syntactic well regimentedness allows us to conclude that since the terms  $r(s^n(t))$  are all orthographically definite, they are also semantically definite. Hence we have

$$(3.62) \quad \iota_G(r(s^n(t))) = \langle \vec{u}\vec{y}\vec{v}\vec{z}\vec{w}, f(g^n(p)) \rangle \in L$$

**Example 36.** The assumption of the syntactic well regimentedness cannot be dropped. Here is an example. Let  $E := v^*$ . According to [Thue, 1914] there is an infinite word  $w_0w_1w_2 \cdots$  over  $\{a, b, c\}$  such that no finite subword is immediately repeated. Let  $M := \{w_0w_1 \cdots w_{n-1} : n \in \mathbb{N}\}$ . Our language is  $\{\langle v^n, w_0w_1 \cdots w_{n-1} \rangle : n \in \mathbb{N}\}$ . Here is a CF bigrammar for it:  $\Omega(f_a) = \Omega(f_b) = \Omega(f_c) = 1$  and  $\Omega(p) = 0$ .

The functions are defined as follows:

$$(3.63) \quad \begin{aligned} \mathcal{J}^\varepsilon(p)() &:= \varepsilon & \mathcal{J}^\mu(p)() &:= \varepsilon \\ \mathcal{J}^\varepsilon(f_a)(\vec{x}) &:= \vec{x} \hat{\ } \mathbf{v} & \mathcal{J}^\mu(f_a)(\vec{x}) &:= \begin{cases} \vec{x} \hat{\ } \mathbf{a} & \text{if } \vec{x} \hat{\ } \mathbf{a} \in M \\ \text{undefined} & \text{else} \end{cases} \\ \mathcal{J}^\varepsilon(f_b)(\vec{x}) &:= \vec{x} \hat{\ } \mathbf{v} & \mathcal{J}^\mu(f_b)(\vec{x}) &:= \begin{cases} \vec{x} \hat{\ } \mathbf{b} & \text{if } \vec{x} \hat{\ } \mathbf{b} \in M \\ \text{undefined} & \text{else} \end{cases} \\ \mathcal{J}^\varepsilon(f_c)(\vec{x}) &:= \vec{x} \hat{\ } \mathbf{v} & \mathcal{J}^\mu(f_c)(\vec{x}) &:= \begin{cases} \vec{x} \hat{\ } \mathbf{c} & \text{if } \vec{x} \hat{\ } \mathbf{c} \in M \\ \text{undefined} & \text{else} \end{cases} \end{aligned}$$

Suppose that the assertion of Theorem 3.9 holds for  $L$ . Then with the notation as in the theorem we would have

$$(3.64) \quad \sigma := \langle \vec{u} \vec{y}^2 \vec{v} \vec{z}^2 \vec{w}, f(g^2(p)) \rangle \in L$$

However,  $g(\vec{x}) = \vec{x} \vec{e}$  for some string  $\vec{e}$ ; and  $f(\vec{x}) = \vec{x} \vec{q}$  for some  $\vec{q}$ . So,  $f(g^2(p)) = p \vec{e} \vec{e} \vec{q}$ . By assumption on  $\sigma \notin L$ , since no string can repeat itself in a string from  $M$ .  $\odot$

The success of the previous counterexample rested in the fact that the same syntactic function is split into different semantic functions. I conjecture that if this were not the case the Theorem 3.9 will also hold for  $L$  even if the grammar is not assumed to be syntactically well regimented. I simply conjecture that it can be shown that the grammar has that property anyway. This would constitute a case where the notions of compositionality based on identity of functions might actually be relevant. If compositionality is based on extensional identity of syntactic functions (see Page 83) then Theorem 3.9 might hold without the assumption of syntactic well regimentedness. However, this still awaits proof.

I stress again that the diverse pumping lemmata discussed in the literature can be generalised to interpreted languages in the same way (Ogden's Lemma, the strengthened form of [Manaster-Ramer *et al.*, 1992], the lemmata for simple literal movement grammars, see [Groenink, 1997], and so on). This is simply because they are all based on the identification of constituents, which are meaningful units of the language.

**Exercise 29.** Show how to generate the language of Example 33 using an independent grammar.

**Exercise 30.** Suppose that  $L \subseteq E \times M$  is an unambiguous countable interpreted language. Show that  $L$  is extensionally autonomous. Show that the result holds also if we assume that there is a number  $k$  such that for every  $e \in E$  there are at most  $k$  many  $m$  with  $\langle e, m \rangle \in L$ .

**Exercise 31.** Suppose that  $L$  is a monophone countable interpreted language. Show that  $L$  is extensionally compositional. *Note.* Show that if  $G$  is defined only on the signs from  $L$ ,  $G$  already is extensionally compositional.

**Exercise 32.** Suppose that  $L \subseteq E \times M$  is a countable interpreted language which is a partial bijection between  $E$  and  $M$ . Then  $L$  is independent.

**Exercise 33.** The following exercise points at some algebraic connections. I refer to the Appendix A for basic algebraic concepts. A **subalgebra** of  $\mathfrak{A} = \langle A, I \rangle$  is a pair  $\mathfrak{B} = \langle B, J \rangle$  such that  $B \subseteq A$  and  $J(f) = I(f) \upharpoonright B$ . In the context of partial algebras this means that for all  $\vec{a} \in B^{\Omega(f)}$ ,  $J(f)(\vec{a})$  is defined if and only if  $I(f)(\vec{a})$  is defined, and their value is the same. There is a subalgebra over the set  $B$  exactly if for all  $f \in F$  we have  $f[B^{\Omega(f)}] \subseteq B$ . Now show the following: (a)  $G$  is autonomous if and only if the algebra of exponents is a subalgebra of the algebra of signs; (b)  $G$  is compositional if and only if the algebra of meanings is a subalgebra of the algebra of signs; (c)  $G$  is independent if the algebra of signs is a direct product of the algebra of exponents and the algebra of meanings.

**Exercise 34.** Show that if a bigrammar is independent then the algebra of signs that it generates is a direct product of its algebra of exponents and its algebra of meanings.

## 3.4 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 [Keenan and Stabler, 2001], [Pollard and Sag, 1994], [Mel'cuk, 1993 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. I am inclined to view categories as basically encoding restrictions stemming from partiality (see [Kracht, 2006]). This makes the formulation somewhat more transparent. For example, in a context free grammar rather than making the string concatenation partial we may say that on the level of exponents there is only one function, concatenation, which is not partial; and that the partiality arises in the categories only. It turns out, though, that one needs to be extremely cautious in thinking that the different formulations are exactly the same. Time and again it appears that they are only the same in ‘normal’ circumstances and that counterexamples to their equivalence exist. This section will elaborate on the theme of categories and prove some results only to abandon the theme later. One result is that in case the set of signs contains only finitely many categories they can be eliminated (Theorem 3.11), though we may be forced to pay a price.

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

$$(3.65) \quad H(\gamma) := \langle e, m \rangle$$

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  $(E \times C \times M)^{\Omega(f)} \hookrightarrow (E \times C \times M)$ . A c-grammar is **autonomous** if the exponent of  $\mathcal{J}(f)(\vec{\sigma})$  is strongly independent of the categories and meanings of the input signs; it is **compositional** if the meaning of  $\mathcal{J}(f)(\vec{\sigma})$  is strongly independent of the exponent and category of the input signs. In addition to the notions of autonomy and compositionality we now have a third notion, which I call **categorial autonomy**. It says that the category of  $\mathcal{F}(f)(\vec{\sigma})$  is strongly independent of the exponents and the meanings of the input signs. The grammar is **independence** if it is autonomous, compositional and categorially autonomous. In case of independence we can exchange the grammar for a different

kind of grammar.

**Definition 3.10** A *trigrammar* over  $E \times C \times M$  is a quadruple  $\langle \Omega, \mathcal{J}^\varepsilon, \mathcal{J}^\kappa, \mathcal{J}^\mu \rangle$ , where  $\Omega$  is a signature and  $\mathcal{J}^\varepsilon$  an interpretation of  $\Omega$  in  $E$ ,  $\mathcal{J}^\kappa$  an interpretation of  $\Omega$  in  $C$ , and  $\mathcal{J}^\mu$  an interpretation of  $\Omega$  in  $M$ .

From a trigrammar we form the corresponding c-grammar by putting

$$(3.66) \quad G_\times := \langle \Omega, \mathcal{J}^\varepsilon \star \mathcal{J}^\kappa \star \mathcal{J}^\mu \rangle$$

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

$$(3.67) \quad \begin{aligned} H[L] &:= \{H(\gamma) : \gamma \in L\} \\ &= \{\langle e, m \rangle : \text{there is } c \in C : \langle e, c, m \rangle \in L\} \end{aligned}$$

**Theorem 3.11** 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

$$(3.68) \quad F_n^+ := \{f_{\vec{c}} : f \in F_n, \vec{c} \in C^n\}$$

For example

$$(3.69) \quad \begin{aligned} F_0^+ &= \{f_{\emptyset} : f \in F_0\} \\ F_1^+ &:= \{f_{\langle c \rangle} : f \in F_1, c \in C\} \\ F_2^+ &:= \{f_{\langle c, c' \rangle} : f \in F_2, c, c' \in C\} \end{aligned}$$

As for the signature, we put

$$(3.70) \quad \Omega^+(f_{\vec{c}}) := \Omega(f)$$

We define the actions of the functions over this signature.

$$(3.71) \quad \begin{aligned} \mathcal{J}(f_{\langle c_0, c_1, \dots, c_{n-1} \rangle})(\langle e_0, m_0 \rangle, \langle e_1, m_1 \rangle, \dots, \langle e_{n-1}, m_{n-1} \rangle) \\ := H(\mathcal{C}(f))(\langle e_0, c_0, m_0 \rangle, \langle e_1, c_1, m_1 \rangle, \dots, \langle e_{n-1}, c_{n-1}, m_{n-1} \rangle) \end{aligned}$$

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

$$(3.72) \quad \mathcal{J}(f_{\vec{c}})(H(\sigma_0), H(\sigma_1), \dots, H(\sigma_{n-1})) := H(\mathcal{C}(f)(\sigma_0, \sigma_1, \dots, \sigma_{n-1}))$$

Here the left hand side is defined if and only if 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  $f^+ := f_{\vec{c}}$ . Now,  $\iota_K(f^+) = H(\iota_G(f))$ , by construction. Inductive case.  $t = f s_0 s_1 \dots s_{n-1}$ .  $\Omega(f) = n > 0$ . 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

$$(3.73) \quad \begin{aligned} \iota_K(t^+) &= \mathcal{J}(f_{\vec{c}})(\iota_K(s_0^+), \iota_K(s_1^+), \dots, \iota_K(s_{n-1}^+)) \\ &= H(\mathcal{C}(f)(\langle e_0, c_0, m_0 \rangle, \dots, \langle e_{n-1}, c_{n-1}, m_{n-1} \rangle)) \\ &= H(\mathcal{C}(f)(\iota_G(s_0), \iota_G(s_1), \dots, \iota_G(s_{n-1}))) \\ &= H(\iota_G(t)) \\ &= H(\sigma) \end{aligned}$$

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:

$$(3.74) \quad (f_{\vec{c}} s_0 \dots s_{\Omega(f)-1})^- := f s_0^- s_1^- \dots s_{\Omega(f)-1}^-$$

In particular,  $(f_{\vec{c}})^- = f$ . 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 definition. 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 (3.72):

$$(3.75) \quad \begin{aligned} H(\iota_G(t^-)) &= H(\iota_G(f s_0^- \dots s_{n-1}^-)) \\ &= H(\mathcal{C}(f)(\iota_G(s_0^-), \dots, \iota_G(s_{n-1}^-))) \\ &= \mathcal{J}(f_{\vec{c}})(H(\iota_G(s_0^-)), H(\iota_G(s_1^-)), \dots, H(\iota_G(s_{n-1}^-))) \\ &= \mathcal{J}(f_{\vec{c}})(\iota_K(s_0), \iota_K(s_1), \dots, \iota_K(s_{n-1})) \\ &= \iota_K(t) \end{aligned}$$

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, mostly only a finite number of them is needed in a particular grammar. 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  $m$ , if it is a verb it means something else, say  $\hat{m}$ . Thus, dictionaries use categories; they say that the language contains two signs:  $\langle \text{leak}, n, m \rangle$  and  $\langle \text{leak}, v, \hat{m} \rangle$ . For example, according to the Shorter Oxford English Dictionary ([Onions, 1973]), “leak” as a verb means: “1. to pass (*out, away, forth*) by a leak or leakage. 2. To let fluid pass in or out through a leak.” The noun has this meaning “1. A hole or fissure in a vessel containing or immersed in a fluid, which lets the fluid pass in or out of the vessel [...] 2. action of leaking or leakage.” These two meanings are clearly distinct. The latter is a physical object (hole) while the former is a process.

If we eliminate the categories, we are left with the signs  $\langle \text{leak}, m \rangle$  and  $\langle \text{leak}, \hat{m} \rangle$ . It seems that vital information is lost, namely that /leak/ means  $m$  *only if it is a noun*, and likewise that it means  $\hat{m}$  *only if it is a verb*. On the other hand, we still know that /leak/ means  $m$  and  $\hat{m}$ . If we perform the construction above, the

following will happen. The function that forms the past tense applies to the sign  $\langle \text{leak}, v, \hat{m} \rangle$  but not to the sign  $\langle \text{leak}, n, m \rangle$ . It is the interpretation of some mode  $f$ . This mode is now replaced among other by a mode  $f_v$ , which takes as input only the sign  $\langle \text{leak}, \hat{m} \rangle$  and forms the sign  $\langle \text{leaked}, \text{past}'(\hat{m}) \rangle$ . It is not defined on  $\langle \text{leak}, m \rangle$ . 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  $\langle \vec{x}, m \rangle$  where  $\vec{x}$  is a root and  $m$  the verbal meaning of that root. It is undefined on  $\langle \vec{y}, m \rangle$  if  $\vec{y}$  cannot be a verbal root or otherwise does not have the meaning  $m$ ; it is not defined on  $\langle \vec{x}, \hat{m} \rangle$  if  $\hat{m}$  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, which employs something that I have dubbed categorial well-regimentation. 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.

As I just said, eliminating categories might come at a price. For example, we might lose compositionality of the grammar. 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 3.5 almost verbatim. The following example now shows that compositionality and autonomy can be lost under reduction.

**Example 37.** Our example is based on the grammar of Example 33. We introduce a set  $C = \{o, p, \}$  of categories. For any given triple  $\langle e, c, m \rangle$  we define



$$(3.76) \quad \begin{aligned} \mathcal{K}(f_1)(\langle e, c, m \rangle) &:= \begin{cases} \langle e \hat{\ } a, p, m + 1 \rangle & \text{if } c = p \\ \text{undefined} & \text{else} \end{cases} \\ \mathcal{K}(f_2)(\langle e, c, m \rangle) &:= \begin{cases} \langle e \hat{\ } a, o, m \rangle & \text{if } c = p \\ \text{undefined} & \text{else} \end{cases} \end{aligned}$$

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 33.

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



Nevertheless, it is also possible to establish a positive result. Let  $L$  be a language. Say that it allows to **guess categories** if the following holds. There are functions  $p : E \rightarrow \wp(C)$  and  $q : M \rightarrow \wp(C)$  such that if  $\langle e, c, m \rangle \in L$  then  $p(e) \cap q(m) = \{c\}$  and that if  $\langle e, c, m \rangle \notin L$  then  $p(e) \cap q(m) = \emptyset$ . This means that if  $e$  and  $m$  are given then  $c$  is unique; and moreover, what can be inferred from  $e$  by itself and by  $m$  itself is enough to guess  $c$ .

**Proposition 3.12** *Let  $L$  be a independent  $c$ -language that allows to guess categories. Suppose further than  $L$  has only finitely many categories. Then  $H[L]$  is independent.*

**Proof.** Let  $p : E \rightarrow \wp(C)$  and  $q : M \rightarrow \wp(C)$  be the guessing functions. Let  $G$  be an independent  $c$ -grammar for  $L$ . By assumption, for every mode  $f$  there are three functions  $f^\varepsilon$ ,  $f^\kappa$  and  $f^\mu$  such that

$$(3.77) \quad \begin{aligned} \mathcal{J}(f)(\langle e_0, c_0, m_0 \rangle, \dots, \langle e_{n-1}, c_{n-1}, m_{n-1} \rangle) \\ = \langle f^\varepsilon(e_0, \dots, e_{n-1}), f^\kappa(c_0, \dots, c_{n-1}), f^\mu(m_0, \dots, m_{n-1}) \rangle \end{aligned}$$

Proceed as in the proof of Theorem 3.11. We create modes of the form  $f_{\vec{c}}$ , where  $\vec{c}$  is a sequence of categories of length  $\Omega(f)$ . Pick an  $n$ -ary mode. If  $n = 0$  and  $\mathcal{J}(F)() = \langle e, c, m \rangle$  let  $\mathcal{J}(f_{\vec{c}})() := \langle e, m \rangle$ . Now suppose that  $n > 0$ . For each  $n$ -ary

sequence of elements from  $C$  we introduce a new mode  $f_{\vec{c}}$ . We set

$$(3.78) \quad f_{\vec{c}}^{\varepsilon}(e_0, \dots, e_{n-1}) := \begin{cases} f^{\varepsilon}(e_0, \dots, e_{n-1}) & \text{if for every } i < n: c_i \in p(e_i) \\ & \text{and } f^{\kappa}(\vec{c}) \text{ is defined} \\ \text{undefined} & \text{else} \end{cases}$$

Likewise we put

$$(3.79) \quad f_{\vec{c}}^{\mu}(m_0, \dots, m_{n-1}) := \begin{cases} f^{\mu}(m_0, \dots, m_{n-1}) & \text{if for every } i < n: c_i \in q(m_i) \\ & \text{and } f^{\kappa}(\vec{c}) \text{ is defined} \\ \text{undefined} & \text{else} \end{cases}$$

This defines the grammar  $G^+$  over the signature  $\Omega^+$ . We show the following claim by induction over the length of the term: (a) if  $\langle e, m \rangle$  is the value of a term  $t$  of length  $n$  then for the unique  $c$  such that  $\langle e, c, m \rangle \in L$ ,  $\langle e, c, m \rangle$  is the value of  $t^-$ ; (b) if  $\langle e, c, m \rangle$  is the value of a term  $t$  of length  $n$  then  $\langle e, m \rangle$  is the value of some term  $u$  such that  $u^- = t$ . This will then establish the claim. Notice first that (a) is straightforward by construction, so we need to establish (b). For length 0 the (b) is certainly true. Now let  $t = f(u_0, \dots, u_{n-1})$ , where  $n = \Omega(f)$ , and let  $\langle e_i, m_i \rangle$ ,  $i < n$ , be the value of  $u_i$ . Note right away that by assumption on  $L$  there can be only one such sequence and hence the set is either empty (no new sign generated) or contains exactly one member (by independence of the modes). Suppose first that for some  $j < n$  there is no  $c$  such that  $\langle e_j, c, m_j \rangle \in L$ . Thus  $p(e_j) \cap q(m_j) = \emptyset$ . Then for every sequence  $\vec{c}$  either  $f_{\vec{c}}^{\varepsilon}(e_0, \dots, e_{n-1})$  or  $f_{\vec{c}}^{\mu}(m_0, \dots, m_{n-1})$  is undefined. Hence none of the functions  $\mathcal{J}(f_{\vec{c}})$  are applicable on this input. Now suppose that for every  $i$  there is a  $g_i$  such that  $\langle e_i, g_i, m_i \rangle \in L$ . We have terms  $u_i^+$  such that  $\langle e_i, g_i, m_i \rangle$  is the value of  $u_i^+$  for  $i < n$ . Then for  $\vec{g} := \langle g_0, \dots, g_{n-1} \rangle$  both  $f_{\vec{g}}^{\varepsilon}(e_0, \dots, e_{n-1})$  and  $f_{\vec{g}}^{\mu}(m_0, \dots, m_{n-1})$  are defined and they equal  $f^{\varepsilon}(e_0, \dots, e_{n-1})$  and  $f^{\mu}(m_0, \dots, m_{n-1})$ , respectively. Since  $f^{\kappa}(g_0, \dots, g_{n-1})$  is also defined (by definition of the functions  $f_{\vec{g}}^{\varepsilon}$  and  $f_{\vec{g}}^{\mu}$ ) the following value exists

$$(3.80) \quad \langle f^{\varepsilon}(e_0, \dots, e_{n-1}), f^{\kappa}(g_0, \dots, g_{n-1}), f^{\mu}(m_0, \dots, m_{n-1}) \rangle$$

This is the value of  $f_{\vec{g}}(u_0^+, \dots, u_{n-1}^+)$ , as is easily seen. (If  $\vec{c} \neq \vec{g}$  then either of the functions  $f_{\vec{c}}^{\varepsilon}(e_0, \dots, e_{n-1})$  and  $f_{\vec{c}}^{\mu}(m_0, \dots, m_{n-1})$  is undefined.)  $\square$

We close this section by some considerations concerning linguistic theories. First, the notion of a grammar as opposed to a bigrammar 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, in a grammar a term is semantically definite if and only if it is orthographically definite. It has a semantics if and only if it has an exponent. By using bigrammars we make these two notions independent. 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 be blamed 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 is no separate level of categories. Technically, the syntax operates before semantics. Syntax operates 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.

### 3.5 Weak and Strong Generative Capacity

Say that two CFGs  $G$  and  $G'$  are *weakly equivalent* if they generate the same string set; and that they are *strongly equivalent* if they assign the same structure to the strings. The question arises what we think to be the structure of the sentence. It turns out that “same structure” depends on personal conviction. It could be, for example, identical topology over the string, or identical tree structure, so that only relabelling is allowed. (See [Miller, 1999] for an excellent discussion.) Typically, it is assumed that structure means tree structure. To say that a language is strongly context free is to assume that the language is given as a set of labelled (ordered)

trees. It is not enough to just consider sets of strings.

In standard linguistic literature it is assumed that syntactic structure is independent of semantic structure. Of course this is an illusion, for all tests assume that when we manipulate certain sentences syntactically we are also manipulating their semantics. For example, when we consider whether /can/ is a noun and we coordinate it with, say, /tray/ to get /can and tray/, we are assuming that we are dealing with it under the same semantics that we have chosen initially (/can/ in the sense of metal object, not the auxiliary). And this should show in the semantics of the coordinate expression. Hence, no syntactic test really can be performed without a semantics. Hence, we shall in this section pursue a different route to “structure”, namely this. We shall explore the idea that structure is in fact epiphenomenal, driven by the need to establish a compositional grammar for the language.

We have defined the associated string language  $\varepsilon[L]$  of an interpreted language to be the set of all strings that have a meaning in  $L$ . We can likewise define for a grammar  $G$  the associated string grammar  $G^\varepsilon$  to consist just of the functions  $f^\varepsilon$  for  $f \in F$ . Since  $f^\varepsilon$  may depend on the meanings of the input signs, this makes immediate sense only for a bigrammar. Even in that case, however, it may happen that  $L(G^\varepsilon) \neq \varepsilon[L]$  precisely because there might be terms which are orthographically definite but not semantically definite. (In general, only  $\varepsilon[L] \subseteq L(G^\varepsilon)$  holds.)

Recall from previous discussions that in grammars the domain of  $f^\mu$  and  $f^\varepsilon$  is identical. In that case some of the distinctions that are of interest in this section cannot be made, such as the distinction between weak dependency of  $f^\varepsilon$  on exponents and the weak dependency of  $f^\mu$  on the exponents. Therefore, in this chapter we shall discuss bigrammars, and not grammars. Recall also from Section 2.3 the discussion of context freeness. There we have define context freeness of a string grammar intrinsically. The results in this section use the term the “context free” in this sense. The results often are more general, applying to concatenative grammars as well. I occasionally point out where results can be generalised.

**Definition 3.13** *Let  $L$  be an interpreted language and  $\mathcal{C}$  a class of string grammars.  $L$  is **weakly**  $\mathcal{C}$  if the associated string language  $\varepsilon[L]$  has a grammar in  $\mathcal{C}$ .  $L$  is  $\mathcal{C}$  if it has a weakly autonomous bigrammar whose associated string grammar is in  $\mathcal{C}$ .  $L$  is **autonomously**  $\mathcal{C}$  if it has a strongly autonomous bigrammar whose associated string grammar is in  $\mathcal{C}$ .*

**Example 38.** An example of an interpreted language that is weakly but not autonomously CF. Let

$$(3.81) \quad L := \{\langle a^n, i \rangle : n \in \mathbb{N}, i < 2^{2^n}\}$$

Given a string  $\vec{x}$  of length  $n$  the number of terms that unfold to  $\vec{x}$  is exponential in  $n$ . This means that there is a number  $p$  such that if  $|\vec{x}| = n$  then the number of parses is bounded by  $2^{pn}$ , provided that  $n$  exceeds some number  $k$ . This means that the number of meanings for the string  $\vec{x}$  cannot exceed  $2^{pn}$ , if  $k < n$ . However, in  $L$   $\vec{x}$  has  $2^{2^n}$  meanings, and for all  $n$  such that  $2^n > p$  we have  $2^{2^n} > 2^{pn}$ .  $\odot$

**Theorem 3.14** *Let  $L$  be unambiguous. Then if  $L$  is weakly  $\mathcal{C}$  it is also autonomously  $\mathcal{C}$ .*

**Proof.** By assumption, there is a function  $f : E \rightarrow M$  such that  $\langle e, m \rangle \in L$  iff  $m = f(e)$  (in set theory,  $L$  is that function  $f$ ). Now with the grammar  $G = \langle \Omega, \mathcal{J} \rangle$  for  $\varepsilon[L]$  in hand, define  $G^+$  as follows. Let  $\Omega(f) = n$ . Then put

$$(3.82) \quad \mathcal{J}^\varepsilon(f)(\langle e_0, m_0 \rangle, \dots, \langle e_{n-1}, m_{n-1} \rangle) := \mathcal{J}(f)(e_0, \dots, e_{n-1})$$

$$(3.83) \quad \mathcal{J}^\mu(f)(\langle e_0, m_0 \rangle, \dots, \langle e_{n-1}, m_{n-1} \rangle) := f(\mathcal{J}(f)(e_0, \dots, e_{n-1}))$$

The bigrammar  $G^+ := \langle G, \mathcal{J}^\varepsilon, \mathcal{J}^\mu \rangle$  is obviously strongly autonomous. Moreover, it generates  $L$ . For if it generates  $\langle e, m \rangle$  then  $m = f(e)$ . Moreover, by construction  $e \in L(G) = \varepsilon[L]$ . Hence  $\langle e, m \rangle \in L$ . If  $G^+$  does not generate  $e$  then either  $m \notin f(e)$ , so  $\langle e, m \rangle \notin L$ , or  $e \notin L(G)$  and so  $e \notin \varepsilon[L]$ , which means that  $\langle e, m \rangle \notin L$ , no matter what  $m$  is.  $\square$

We can strengthen this as follows.

**Theorem 3.15** *Let  $L$  be unambiguous and monophone. Then if  $L$  is weakly  $\mathcal{C}$  it is also strongly  $\mathcal{C}$ .*

**Proof.** By the previous theorem,  $L$  is autonomous. So we already have  $f_*^\varepsilon$  independent of the meanings. The art is in defining the semantic functions. By assumption, there is a partial injection  $\pi : E \rightarrow M$  such that  $L = \{\langle e, \pi(e) \rangle : e \in \varepsilon[L]\}$ . With the help of this bijection put

$$(3.84) \quad f_*^\mu(m_0, \dots, m_{\Omega(f)-1}) := \pi(f_*^\varepsilon(\pi^{-1}(m_0), \dots, \pi^{-1}(m_{\Omega(f)-1})))$$

This defines a grammar that is compositional.  $\square$

Notice that most interesting languages fail to be monophonous. Hence the notions based on string grammars are not as interesting as they appear, despite the fact that weak  $\mathcal{C}$  does not imply strong  $\mathcal{C}$ . A more interesting notion is provided by restricting the set of grammars to independent bigrammars. In this case the semantic functions are required to act independently of the string functions. This means that the added semantic functions must give a unique value independently of the strings. It is however possible to tailor the *domain* of the semantic functions using the exponents. If the latter option is unavailable, we talk of superstrong generative capacity. It means that the semantic functions do not need to see the exponents nor even to know when they should be undefined.

**Definition 3.16** *Let  $L$  be a language and  $\mathcal{C}$  a class of string grammars.  $L$  is **strongly**  $\mathcal{C}$  if it has a weakly independent bigrammar whose associated string grammar is in  $\mathcal{C}$ .  $L$  is **superstrongly**  $\mathcal{C}$  if it has an independent bigrammar whose associated string grammar is in  $\mathcal{C}$ .*

We shall see below an example of a language that is weakly CF but neither superstrongly nor strongly CF and an example of a language that is strongly CF but not superstrongly CF. Notice that by definition CF grammars are strongly autonomous, so the distinction between strong and superstrong turns on the possibility to have a weakly compositional or compositional CF grammar, respectively.

**Example 39.** (See also [Janssen, 1997].) This example shows that weakly equivalent grammar classes may not be strongly equivalent. A CFG  $G$  is **left regular** if it only has rules of the form  $A \rightarrow Bx$  or  $A \rightarrow x$ ,  $A$  and  $B$  nonterminals and  $x$  a terminal symbol.  $G$  is **right regular** if it only has rules of the form  $A \rightarrow xB$  or  $A \rightarrow x$ ,  $A$  and  $B$  nonterminals and  $x$  a terminal symbol. Let  $\mathcal{CL}$  be the class of left regular grammars and  $\mathcal{CR}$  the class of right regular grammars. The language we look at is the language of binary strings and their ordinary denotations:  $A := \{0, L\}$ . For nonempty  $\vec{x} \in A^*$  we put

$$\begin{aligned}
 n(0) &:= 0 \\
 n(L) &:= 1 \\
 n(\vec{x}0) &:= 2n(\vec{x}) \\
 n(\vec{x}L) &:= 2n(\vec{x}) + 1
 \end{aligned}
 \tag{3.85}$$

Finally,

$$(3.86) \quad L := \{\langle \vec{x}, n(\vec{x}) \rangle : \vec{x} \in A^+\}$$

This language is weakly left regular and weakly right regular. It is super strongly left regular, but not strongly right regular. Here is a left regular strongly autonomous bigrammar.  $F := \{f_0, f_1, f_2, f_3\}$ ,  $\Omega(f_0) = \Omega(f_1) = 0$ ,  $\Omega(f_2) = \Omega(f_3) = 1$ .

$$(3.87) \quad \begin{aligned} \mathcal{J}(f_0)() &:= \langle 0, 0 \rangle \\ \mathcal{J}(f_1)() &:= \langle L, 1 \rangle \\ \mathcal{J}(f_2)(\langle \vec{x}, n \rangle) &:= \langle \vec{x}^{\wedge} 0, 2n \rangle \\ \mathcal{J}(f_3)(\langle \vec{x}, n \rangle) &:= \langle \vec{x}^{\wedge} L, 2n + 1 \rangle \end{aligned}$$

There is however no independent left regular bigrammar for this language. Suppose to the contrary that there is such a bigrammar. It has zeroary functions (to reflect the terminal rules) and unary functions. The latter reflect the nonterminal rules. Hence, they must have the form

$$(3.88) \quad f^e(\langle \vec{x}, n \rangle) = \vec{y}^{\wedge} \vec{x}$$

where  $\vec{y}$  is a single symbol.

I now give a combinatorial argument that is worth remembering. Consider the following strings:

$$(3.89) \quad L0, L00, L000, L0000, \dots$$

These strings must be obtained by adding /L/ to a string consisting of zeroes. We do not know which function is responsible for adding the /L/ in the individual cases (we may have any number of modes) but what we do know is that there is one mode  $f$  such that  $\mathcal{J}(f)$  creates two many of them, say /L000/ and /L0000000/. By definition, it creates them from the strings /000/ and /0000000/, respectively. Now, these strings have the same meaning, namely 0. If the grammar is compositional,  $f^\mu$  is independent of the exponent. However, we must now have  $f^\mu(0) = 8$ , as well as  $f^\mu(0) = 256$ , a contradiction.

$$(3.90) \quad \begin{aligned} \mathcal{J}(\langle 000, 0 \rangle) &= \langle L000, 8 \rangle &= \langle f^e(000), f^\mu(0) \rangle \\ \mathcal{J}(\langle 0000000, 0 \rangle) &= \langle L0000000, 256 \rangle &= \langle f^e(0000000), f^\mu(0) \rangle \end{aligned}$$



This argument is pretty robust, it precludes a number of strategies. For example, making syntactic or semantic functions partial will obviously not improve matters.

The example is useful also because it shows the following. Suppose that  $\mathcal{C}$  and  $\mathcal{D}$  are classes of string grammars, such that every string language that is  $\mathcal{C}$  is also  $\mathcal{D}$ . Then it does not necessarily hold that a language that is superstrongly  $\mathcal{C}$  is also superstrongly  $\mathcal{D}$ . For in the above example, we have two classes of grammars that generate the same set of string languages, but they are not identical when it comes to interpreted languages.

The proof in the previous example is somewhat less satisfying since CFGs also use categories, though it works in that case as well. In order to include categories we have to switch to c-languages. We shall not introduce special terminology here to keep matters simple. Basically, if  $L$  is a language of c-signs it is weakly CF if the associated string language is CF. It is CF if there is an independent c-grammar for it whose string *and* category part taken together is CF.

**Example 40.** We continue Example 39. Given the same language  $L$  we show that there is no right regular c-language  $M$  whose projection to  $A^* \times M$  is  $L$ . This is to say, allowing *any* classification  $M$  of string-meaning pairs into finitely many categories, there is no independent right regular c-grammar for  $M$ . The argument is basically the same. We look at unary functions. If  $f$  is unary, it has the form

$$(3.91) \quad \mathcal{J}(\langle \vec{x}, \gamma, n \rangle) = \langle f_*^\varepsilon(\vec{x}), f_*^k(\gamma), f_*^\mu(n) \rangle$$

for some  $f_*^\varepsilon$ ,  $f_*^k$  and  $f_*^\mu$ . Furthermore,  $f_*^\varepsilon(\vec{x}) = \vec{y} \hat{\ } \vec{x}$ . Look at the signs  $\sigma_p := \langle \text{LO}^p, \gamma_p, 2^p \rangle$ . Let  $t_p$  be the analysis term of  $\sigma_p$ . (Being a left regular grammar, we can assume each sign has at most one analysis term.) Either  $t_p = f$  for some zeroary  $f$ , or  $t_p = fs_p$  for some unary  $f$ . In the latter case,  $f_*^\varepsilon(\vec{x}) = \text{LO}^k \hat{\ } \vec{x}$  for some  $k$  which depends only on  $f$  and so  $s_p$  unfolds to  $\langle \text{O}^{p-k}, \delta_p, 0 \rangle$ . Now we look at  $f_*^\mu$ . We have  $f_*^\mu(0) = 2^p$ . It follows that if  $q \neq p$  then  $t_q$  does not have the form  $fs$ . There are however only finitely many functions.  $\odot$

**Example 41.** An example to show that strong and superstrong CF languages are distinct. Consider the number expressions of English. We may for simplicity assume that the highest simple numeral is /million/. To keep this example small we add just the following words: /one/, /ten/, /hundred/, /thousand/. It will be



easy to expand the grammar to the full language. Number expressions are of the following kind: they are nonempty sequences

$$(3.92) \quad \vec{x}_0(\text{million}_\perp)^{p_0} \wedge \vec{x}_1(\text{million}_\perp)^{p_1} \wedge \dots \wedge (\vec{x}_{n-1}\text{million}_\perp)^{p_{n-1}}$$

where  $p_0 > p_1 > \dots > p_{n-1}$ , and the  $\vec{x}_i$  are expressions not using /million/, which are nonempty sequences of the following form.

$$(3.93) \quad ((\text{one}_\perp \mid \text{ten}_\perp \mid \text{one\_hundred}_\perp)\text{thousand}_\perp)? \\ (\text{one}_\perp \mid \text{ten}_\perp \mid \text{one\_hundred}_\perp)?$$

This language is not weakly CF. It does not satisfy the pumping lemma (see Exercise 36). It can therefore not be superstrongly CF. However, it is strongly CF. Here is a grammar for it. Call a **block** an expression containing /million/ only at the end. Say that  $\vec{x}$  is **m-free** if it does not contain any occurrences of /million/, and that it is **t-free** if it does not contain any occurrences of /million/ and /thousand/. The grammar is given in Figure 3.5. It has two modes of composition: ‘additive’ concatenation and ‘multiplicative’ concatenation. Since the language is unambiguous, we can formulate a bigrammar using string functions that are total, and semantic functions that are partial. Now let  $A(\vec{x}, \vec{y}, m, n)$  hold if and only if either (a)  $\vec{x}$  is a block and  $m > n$  or (b)  $\vec{x}$  is m-free but not t-free and  $\vec{y}$  is t-free. Let  $B(\vec{x}, \vec{y}, m, n)$  if and only if either (a)  $\vec{x}$  is a block and  $\vec{y} = \text{million}$  or (b)  $\vec{x} = \text{one}$  and  $\vec{y} = \text{hundred, thousand}$  or (c)  $\vec{x} = \text{one\_hundred}$  and  $\vec{y} = \text{thousand}$ . (See Figure 3.5.) Then put

$$(3.94) \quad \begin{aligned} a^\varepsilon(\langle \vec{x}, m \rangle, \langle \vec{y}, n \rangle) &:= \vec{x} \wedge \vec{y} \\ a^\mu(\langle \vec{x}, m \rangle, \langle \vec{y}, n \rangle) &:= \begin{cases} m + n & \text{if } A(\vec{x}, \vec{y}, m, n) \\ \text{undefined} & \text{else} \end{cases} \\ m^\varepsilon(\langle \vec{x}, m \rangle, \langle \vec{y}, n \rangle) &:= \vec{x} \wedge \vec{y} \\ m^\mu(\langle \vec{x}, m \rangle, \langle \vec{y}, n \rangle) &:= \begin{cases} mn & \text{if } B(\vec{x}, \vec{y}, m, n) \\ \text{undefined} & \text{else} \end{cases} \end{aligned}$$

Thus, the semantic functions are weakly independent of the exponents, but not strongly independent.

Variations can be played on this theme. First, if we introduce the word /zero/ and allow the use of expressions /zero\_million\_\perp^k/ then the semantic condition “ $m > n$ ” in  $A(\vec{x}, \vec{y}, m, n)$  must be replaced by a syntactic condition involving the

Figure 3.5: Number Names

$$\begin{aligned}
\mathcal{J}(f_0)() &:= \langle \text{one}, 1 \rangle \\
\mathcal{J}(f_1)() &:= \langle \text{ten}, 10 \rangle \\
\mathcal{J}(f_2)() &:= \langle \text{hundred}, 100 \rangle \\
\mathcal{J}(f_3)() &:= \langle \text{thousand}, 1000 \rangle \\
\mathcal{J}(f_4)() &:= \langle \text{million}, 1,000,000 \rangle \\
\mathcal{J}(a)(\langle \vec{x}, m \rangle, \langle \vec{y}, n \rangle) &:= \begin{cases} \langle \vec{x} \frown \vec{y}, m+n \rangle & \text{if } \vec{x} \text{ is a block and } m > n \\ & \text{or } \vec{x} \text{ m-free but not t-free,} \\ & \text{and } \vec{y} \text{ is t-free} \\ \text{undefined} & \text{else} \end{cases} \\
\mathcal{J}(m)(\langle \vec{x}, m \rangle, \langle \vec{y}, n \rangle) &:= \begin{cases} \langle \vec{x} \frown \vec{y}, mn \rangle & \text{if } \vec{x} \text{ is a block, and } \vec{y} = \text{million} \\ & \text{or } \vec{x} = \text{one, and} \\ & \vec{y} = \text{hundred, thousand} \\ & \text{or } \vec{x} = \text{one\_hundred,} \\ & \vec{y} = \text{thousand} \\ \text{undefined} & \text{else} \end{cases}
\end{aligned}$$

number  $k$ . In that case it seems more consistent to say that the semantic functions are total while the syntactic functions are restricted, and so the language is not really CF.  $\odot$

**Example 42.** Here is another example, see [Radzinski, 1990]. In Chinese, yes-no questions are formed by iterating the VP. I reproduce the syntax of Chinese in English. To ask whether John went to the shop you say

(3.95) John went to the shop not went to the shop?

The recipe is this. Given a subject  $\vec{x}$ , and a VP  $\vec{y}$ , the yes-no question is formed like this

(3.96)  $\vec{x} \_ \vec{y} \_ \text{not} \_ \vec{y}?$

The data for Chinese is not without problems, but I shall ignore the empirical complications here and pretend that the above characterisation is exact. One analysis proceeds via copying. An alternative analysis is the following. Observe that in Chinese, disjunctive statements are formed like this:

$$(3.97) \quad \vec{x} \_ \vec{y} \_ \vec{z}.$$

In particular, a disjunction between  $\vec{y}$  and *not*  $\vec{z}$  is expressed like this:

$$(3.98) \quad \vec{x} \_ \vec{y} \_ \text{not} \_ \vec{z}.$$

In this case it is required that  $\vec{z} \neq \vec{y}$ . This suggests that we may also form the yes-no question by concatenation, which however is partial. It is possible to construct a weakly CF bigrammar, but not a strongly CF one.  $\otimes$

I shall now return to the question whether ambiguity can be removed from a language. The question is whether there is a transform of a language into an unambiguous language and how that affects the possibility of generating it with a given class of grammars. It shall emerge that there are languages which are inherently structurally ambiguous. This means the following. Given a language  $L$  which is unambiguous, every derivation of a given exponent must yield the same meaning. Thus, as one says, all structural ambiguity is *spurious*.

**Definition 3.17** *Let  $G$  be a grammar. A  $G$ -ambiguity is a pair  $(t, t')$  of nonidentical terms such that  $\iota_G(t) = \langle e, m \rangle$  and  $\iota_G(t') = \langle e, m' \rangle$  for some  $e, m$  and  $m'$ . In this case we call  $e$  **structurally ambiguous in  $G$** . The ambiguity  $(t, t')$  is **spurious** if  $m = m'$ . Also,  $(t, t')$  is a **lexical ambiguity**, where  $t \approx_0 t'$ , which is defined as follows:*

$$(3.99) \quad \begin{aligned} & f \approx_0 g \text{ if } \Omega(f) = \Omega(g) = 0 \\ & f s_0 \cdots s_{n-1} \approx_0 f t_0 \approx t_{n-1} \text{ if } n > 0, f = g \text{ and } s_i \approx_0 t_i \text{ for all } i < n \end{aligned}$$

*An ambiguity that is not lexical is called **structural**.*

Alternatively, an ambiguity is a pair  $(t, u)$  where  $t^\varepsilon = u^\varepsilon$ . Let  $L$  be a language. Then define the functional transform of  $L$  in the following way. For  $e$  we put  $e^\circ := \{m : \langle e, m \rangle \in L\}$ .

$$(3.100) \quad L^\S := \{\langle e, e^\circ \rangle : e \in \varepsilon[L]\}$$

The functional transform of  $L$  is such that every  $e$  has exactly one meaning, which is the (nonempty) set of meanings that  $e$  has in  $L$ .

**Example 43.** We let  $A := \{\mathbf{p}, \mathbf{0}, 1, \neg, \wedge, \vee\}$ .  $F := \{f_0, f_1, f_2, f_3, f_4, f_5\}$ ,  $\Omega(f_0) := 0$ ,  $\Omega(f_1) := \Omega(f_2) := \Omega(f_3) := 0$ ,  $\Omega(f_4) := \Omega(f_5) := 2$ . Meanings are sets of functions from  $V := \{\mathbf{0}, 1\}^*$  to  $\{t, f\}$ . We define UBool as the language generated by the following CFG  $G_U$ . For a variable  $\mathbf{p}\vec{x}$ ,  $[\mathbf{p}\vec{x}] = \{\beta : \beta(\vec{x}) = t\}$ . Given  $U = [\mathbf{p}\vec{x}]$ , it is possible to recover  $\vec{x}$ . Given  $U$ , let  $\dagger U$  be the unique  $\vec{x}$  for which  $[\vec{x}] = U$ . The set of all valuations is denoted by Val.

$$\begin{aligned}
\mathcal{J}(f_0)(\mathbf{0}) &:= \langle \mathbf{p}, [\varepsilon] \rangle \\
\mathcal{J}(f_1)(\langle \vec{x}, U \rangle) &:= \langle \vec{x} \mathbf{0}, [(\dagger U) \mathbf{0}] \rangle \\
\mathcal{J}(f_2)(\langle \vec{x}, U \rangle) &:= \langle \vec{x} 1, [(\dagger U) 1] \rangle \\
\mathcal{J}(f_3)(\langle \vec{x}, U \rangle) &:= \langle \neg \vec{x}, \text{Val} - U \rangle \\
\mathcal{J}(f_4)(\langle \vec{x}, U \rangle, \langle \vec{y}, V \rangle) &:= \langle \vec{x} \wedge \vec{y}, V \cap U \rangle \\
\mathcal{J}(f_5)(\langle \vec{x}, U \rangle, \langle \vec{y}, V \rangle) &:= \langle \vec{x} \vee \vec{y}, V \cup U \rangle
\end{aligned}
\tag{3.101}$$

Notice that this language is like natural language in being highly ambiguous: there are no brackets. Thus, the expression  $/\neg \mathbf{p}\mathbf{0}\wedge \mathbf{p}/$  can be read in two ways: it has analysis terms  $f_3 f_4 f_1 f_0 f_0$ , with negation having scope over conjunction, and  $f_4 f_3 f_1 f_0 f_0$ , with conjunction having scope over negation. Clearly, the meanings are different.  $\odot$

Let us now try to see whether we can define a CFG for UBool<sup>s</sup>. We shall keep the string part of  $G_U$  from Example 43. Look at the strings  $/\mathbf{p}\vec{x}\wedge\neg\mathbf{p}\vec{x}/$ , where  $\vec{x} \in \{\mathbf{0}, 1\}^*$ . As they are uniquely readable and they have no satisfying valuation, their meaning in UBool<sup>s</sup> is  $\{\emptyset\}$ . On the other hand,  $/\mathbf{p}\vec{x}\wedge\neg\mathbf{p}\vec{x}\vee\mathbf{p}\vec{y}/$  has three analyses corresponding to the following bracketed strings:

$$\tag{3.102} \quad /((\mathbf{p}\vec{x}\wedge(\neg\mathbf{p}\vec{x}))\vee\mathbf{p}\vec{y})/, /(\mathbf{p}\vec{x}\wedge(\neg(\mathbf{p}\vec{x}\vee\mathbf{p}\vec{y})))/, /(\mathbf{p}\vec{x}\wedge((\neg\mathbf{p}\vec{x})\vee\mathbf{p}\vec{y}))/$$

Thus the meaning is  $\{[\vec{y}], [\vec{x}] \cap [\vec{y}], \emptyset\}$ . Let us now look at one particular analysis.

$$\begin{aligned}
\tag{3.103} \quad \mathcal{J}(f_5)(\langle \mathbf{p}\vec{x}\wedge\neg\mathbf{p}\vec{x}, \{\emptyset\} \rangle, \langle \mathbf{p}\vec{y}, [\vec{y}] \rangle) \\
= \langle \mathbf{p}\vec{x}\wedge\neg\mathbf{p}\vec{x}\vee\mathbf{p}\vec{y}, \{[\vec{y}], [\vec{x}] \cap [\vec{y}], \emptyset\} \rangle
\end{aligned}$$

In this analysis, there are infinitely many results for this pair of inputs, so this is a case of a grammar that cannot be strongly compositional. There is a possibility, though, of making the result undefined for this analysis term. Another analysis is this:

$$\begin{aligned}
\tag{3.104} \quad \mathcal{J}(f_4)(\langle \mathbf{p}\vec{x}, [\vec{x}] \rangle, \langle \neg\mathbf{p}\vec{x}\vee\mathbf{p}\vec{y}, \{(\text{Val} - [\vec{x}]) \cup [\vec{y}], \text{Val} - ([\vec{x}] \cup [\vec{y}])\} \rangle) \\
= \langle \mathbf{p}\vec{x}\wedge\neg\mathbf{p}\vec{x}\vee\mathbf{p}\vec{y}, \{[\vec{y}], [\vec{x}][\vec{y}], \emptyset\} \rangle
\end{aligned}$$

Here, the arguments provide enough information to compute the result. Thus, it is conceivable that an independent grammar exists.

Notice that we have so far only shown that there can be no compositional CF grammar that uses the structure that the formulae ordinarily have. It is not ruled out that some unconventional structure assignment can actually work. In fact, for this language no compositional CF grammars exists. As a warm-up for the proof let us observe the following. Let  $\varphi$  be a formula that is composed from variables and conjunction. Then although  $\varphi$  may be ambiguous, all the ambiguity is spurious: it has one meaning only. It is the set of assignments that make all occurring variables true. Notice additionally that neither the order nor the multiplicity of the variables matters. Thus the following have identical meaning:  $/p\wedge p\wedge p1/$ ,  $/p1\wedge p\wedge p1\wedge p1/$ ,  $/p\wedge p\wedge p1\wedge p1/$ . Next we consider formulae of the form  $\alpha\vee\varphi$ , where  $\alpha$  is a variable, and  $\varphi$  is of the previous form. An example is  $/p\wedge v\wedge p1\wedge p1\wedge p2/$ . We assume that  $\alpha$  does not occur in  $\varphi$  and that all occurrences of the same variable are adjacent. Up to spurious ambiguity this formula has the following bracketing (conjunction binding stronger than disjunction):

$$(3.105) \quad \begin{aligned} & (p\wedge v\wedge p1\wedge p1\wedge p2) \\ & (p\wedge v\wedge p)\wedge p1\wedge p1\wedge p2 \\ & (p\wedge v\wedge p\wedge p1)\wedge p1\wedge p2 \\ & (p\wedge v\wedge p\wedge p1\wedge p1)\wedge p2 \end{aligned}$$

The general form is  $(\alpha \vee \chi) \wedge \rho$ , and its satisfying valuations make either  $\alpha \wedge \rho$  or  $\chi \wedge \rho$  true.  $\alpha$  is a single variable. It is easy to see that it makes no difference whether a variable occurs twice or more, while it may matter whether it occurs once or twice. If  $v$  occurs once, it has a choice to be in  $\chi$  or in  $\rho$ . How often it occurs in either of them does not matter. If  $v$  occurs twice, it may additionally occur both in  $\chi$  and  $\rho$ . However, even in that case there is no difference. Assuming that  $v$  does not occur in  $\alpha$ ,  $\chi$  or  $\rho$ , here are the choices if it occurs just once:

$$(3.106) \quad (\alpha \vee \chi) \wedge v \wedge \rho, (\alpha \vee \chi \wedge v) \wedge \rho$$

Here are the choices if it occurs twice:

$$(3.107) \quad (\alpha \vee \chi) \wedge v \wedge v \wedge \rho, (\alpha \vee \chi \wedge v) \wedge v \wedge \rho, (\alpha \vee \chi \wedge v \wedge v) \wedge \rho.$$

The first reading of (3.107) is the same as the first reading of (3.106), the last reading of (3.107) the same as the last reading of (3.106). The middle reading is

synonymous with the first. (This argument requires  $\chi$  to be nonempty.) For the purpose of the next theorem say that a bigrammar  $\langle \Omega, \mathcal{J}^\varepsilon, \mathcal{J}^\mu \rangle$  is a concatenation bigrammar if  $\langle \Omega, \mathcal{J}^\varepsilon \rangle$  is a concatenation grammar. (Notice that the meaning functions can be partial, too, and that their partiality is not counted in the definition, since we take the string reduct of the grammar.)

**Theorem 3.18** *UBool<sup>§</sup> has no independent concatenation bigrammar. Hence, UBool<sup>§</sup> is not strongly CF and also not superstrongly CF.*

**Proof.** The proof will establish that there is no strongly independent concatenative grammar that has no syncategorematic symbols. We leave the rest of the proof to the reader. The grammar uses the alphabet of the language, the meanings as specified, and a set  $C$  of categories. The functions on the exponents are total. Partiality exists in the semantics. It will emerge from the proof, however, that introducing partiality will not improve the situation. For we shall show that for given  $n$  there is an exponential number of formulae which have to be derived from a polynomially bounded family of formulae via a one step application. This is impossible. If the modes are partial, this remains impossible since it gives us less definite terms not more. Superstrongly CF grammars do not allow any dependency of the meaning on the strings. Thus, for every mode  $f$  and  $\sigma_i = \langle e_i, m_i \rangle$ ,  $i < \Omega(f)$ , we have

$$(3.108) \quad \mathcal{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1}) = \langle f^\varepsilon(e_0, \dots, e_{\Omega(f)-1}), f^\mu(\sigma_0, \dots, \sigma_{\Omega(f)-1}) \rangle$$

Let us look at the following kinds of expressions, where  $V = \mathbf{p}(\mathbf{0} \mid \mathbf{1})^*$  is the set of variables:

$$(3.109) \quad V \vee (V \wedge)^+ V \vee V$$

For ease of understanding, we shall first ignore the internal structure of variables and present them as units. The more concrete structure of our formulae are as follows, in ordinary notation:

$$(3.110) \quad \varphi = p_0 \vee p_2 \wedge p_4 (\wedge p_4) \wedge p_5 (\wedge p_5) \cdots p_{n+3} (\wedge p_{n+3}) \wedge p_3 \vee p_1$$

Let us say that  $\varphi$  has a **cut** at  $i$  if the letter  $p_i$  is repeated twice. Let  $I$  be the set of indices  $i$  such that  $p_i$  occurs in  $\varphi$ ; let  $R$  be a subset of  $I$ . Then by  $\varphi_R$  denote the formula that is like  $\varphi$  having a cut exactly at those  $i$  that are in  $R$ . We show first the following claim.

**Claim.** Let  $R, S \subseteq [4, n + 3] = [4, 5, \dots, n + 3]$ . If  $R \neq S$  then the meaning of  $\varphi_R$  in  $\text{UBool}^{\S}$  is different from that of  $\varphi_S$ .

Let's look at the possible readings of such a formula. Pick a variable  $v = p_i$ . Bracketings are of several forms.

The first set is where the scopes of the disjunctions are nested: we consider the case where the first disjunct takes scope over the second (the other case is dual). (Here,  $\wedge$  binds stronger than  $\vee$ .  $\gamma_1$  may be empty;  $\delta_2$  may not be.)

**(Form 1)**  $(p_0 \vee \gamma_1 \wedge (\gamma_2 \wedge p_i \wedge \delta \vee p_1))$  or  $(p_0 \vee \gamma_1 \wedge (\gamma_2 \wedge p_i \wedge p_i \wedge \delta \vee p_1))$

**(Form 2)**  $(p_0 \vee \gamma \wedge p_i \wedge \delta_1 \wedge (\delta_2 \vee p_1))$  or  $(p_0 \vee \gamma \wedge p_i \wedge p_i \wedge \delta_1 \wedge (\delta_2 \vee p_1))$

**(Form 3)**  $(p_0 \vee \gamma \wedge p_i \wedge (p_i \wedge \delta \vee p_1))$

The two variants of Form (1) and (2) are equivalent. Form (3) is equivalent with Form (2) with  $\delta = \delta_2$ . Let us now consider the case where the scopes of the disjunction signs do not intersect. We get the following list of forms, where it is assumed that  $\gamma$ ,  $\delta_1$  and  $\delta_2$  do not contain  $p_i$ .

**(Form A)**  $(p_0 \vee \gamma \wedge p_i) \wedge \delta_1 \wedge (\delta_2 \vee p_1)$  or  $(p_0 \vee \gamma \wedge p_i \wedge p_i) \wedge \delta_1 \wedge (\delta_2 \vee p_1)$ ;

**(Form B)**  $(p_0 \vee \gamma_1) \wedge \gamma_2 \wedge (p_i \wedge \delta \vee p_1)$  or  $(p_0 \vee \gamma_1) \wedge \gamma_2 \wedge (p_i \wedge p_i \wedge \delta \vee p_1)$ ;

**(Form C)**  $(p_0 \vee \gamma_1) \wedge \gamma_2 \wedge p_i \wedge \delta_1 \wedge (\delta_2 \vee p_1)$  or  $(p_0 \vee \gamma_1) \wedge \gamma_2 \wedge p_i \wedge p_i \wedge \delta_1 \wedge (\delta_2 \vee p_1)$ ;

**(Form D)**  $(p_0 \vee \gamma_1) \wedge \gamma_2 \wedge p_i \wedge (p_i \wedge \delta \vee p_1)$ ;

**(Form E)**  $(p_0 \vee \gamma \wedge p_i) \wedge p_i \wedge \delta_1 \wedge (\delta_2 \vee p_1)$ ; and

**(Form F)**  $(p_0 \vee \gamma \wedge p_i) \wedge (p_i \wedge \delta \vee p_1)$ .

(We allow  $\delta_i$  and  $\gamma_j$  to be empty.) The two variants of Forms (A), (B) and (C) are equivalent. Forms (D), (E) and (F) only exist if the formula has a cut at  $i$ . Thus, it is enough if we show that one of them has no equivalent formula of either of (A), (B) and (C). It is easily seen that Form (D) is equivalent to Form (C) with  $\delta_2 = \delta$ .

Similarly, Form (E) is equivalent to Form (C) with  $\gamma_1 = \gamma$ . Finally, we turn to Form (F):

$$(3.111) \quad \begin{aligned} & (p_0 \vee \gamma \wedge p_i) \wedge (p_i \wedge \delta \vee p_1) \\ & = (p_0 \wedge p_i \wedge \delta) \vee (p_0 \wedge p_1) \vee (\gamma \wedge p_i \wedge p_i \wedge \delta) \vee (\gamma \wedge p_i \wedge p_1) \end{aligned}$$

Form (F) has a disjunct of the form  $p_0 \wedge p_1$ . This is only the case with Forms (1) and (2), (A) with  $\delta_1$  empty, and (B) with  $\gamma_2$  empty. Form (F) implies  $(\neg p_0) \rightarrow \gamma$ , as well as  $(\neg p_1) \rightarrow \delta$ . In Form (1), we therefore must have  $\gamma_1 = \gamma$  and in Form (2)  $\delta_2 = \delta$ . Form (F) implies  $\neg(p_0 \wedge p_1) \rightarrow p_i$ . This is not a consequence of Forms (1) and (2), (A) or (B). Thus, Form (F) is non equivalent to any of the previous forms.

It follows that if the formula has a cut at  $i$ , it has a reading different from the formula obtained by removing this cut by removing one occurrence of  $p_i$ . Now,  $i$  was completely arbitrary. Thus the Claim is established.

Now consider an analysis term of  $\varphi_R$ . The immediate constituents of  $\varphi_R$  cannot contain two disjunction symbols. They can only contain one. In this case, however, the cuts present in  $\varphi_R$  are not reflected in the semantics. To conclude the argument, let us assume that the analysis term of  $\varphi_R$  is  $f s_0 \cdots s_{\Omega(f)-1}$ . We shall look at all possible analysis terms for the  $\varphi_S$ ,  $S \subseteq [4, n+3]$ . We look at (3.108) and count how many meanings we can compose in this way. The syntactic function is total. Let  $k^*$  be the maximal arity of functions and  $p := \text{card } C$  the number of nonterminal symbols. Choose a decomposition into parts; each part has a meaning that is determined just by the subset of  $[i, j] \subseteq [2, n+3]$  of indices for variables that occur in it (and whether or not it contains  $p_0, p_1$ ). For the category there is a choice of  $p$  symbols. The meanings must exhaust the set  $[2, n+3]$ . They can overlap in a single number (since sometimes  $p_i$  can occur twice). There are in total at most  $(2p)^{k^*} \binom{n+2}{k^*-1}$  ways to cut  $\varphi_R$  into maximally  $k^*$  parts of different category and different meaning. The combinations of category and meaning do not depend on  $R$ . We have

$$(3.112) \quad (2p)^{k^*} \binom{n+2}{k^*-1} < (2p(n+2))^{k^*}$$

Out of such parts we must form in total  $2^n$  different meanings to get all the  $\varphi_S$ , using our modes. Assume that we have  $\mu$  modes. If  $n$  is large enough, however,  $\mu(2p(n+2))^{k^*} < 2^n$ .  $\square$

The proof has just one gap and it consists the question of variables. The variables cannot be simple and need to be constructed as well using some modes.



It is not difficult to see that here again just a polynomial number of choices exist, too few to generate the entire number of formulae that are needed. (See also Exercise 37 below.)

There is an interesting further question. Consider in place of the meaning  $e^\circ$  another one; given that meanings are propositions we can form the disjunctions of all the possible meanings.

$$(3.113) \quad \begin{aligned} e^\vee &:= \bigvee \{m : \langle e, m \rangle \in L\} \\ L^\vee &:= \{\langle e, e^\vee \rangle : e \in \varepsilon[L]\} \end{aligned}$$

This leads to the language  $\text{UBool}^\vee$ . It is not clear whether this language is (super)strongly CF.

**Exercise 35.** Prove Theorem 3.14. Prove that the theorem can be strengthened to languages where a string has boundedly many meanings.

**Exercise 36.** The Pumping Lemma says that if a string language  $L$  is CF then there is a number  $k$  such that for every string  $\vec{x} \in L$  of length  $> k$  there is a decomposition  $\vec{x} = \vec{u}\vec{y}\vec{v}\vec{z}\vec{w}$  such that for all  $n$  (including  $n = 0$ ):  $\vec{u}\vec{y}^n\vec{v}\vec{z}^n\vec{w} \in L$ . (See Section 3.4.) Show that the language in Example 41 does not satisfy the Pumping Lemma.

**Exercise 37.** Look again at  $\text{UBool}$ . Call a **formula** a string of  $\varepsilon[\text{UBool}]$  that contains  $/p/$ . (The remaining strings are **indices**.) Subformulae are (occurrences) of formulae in the ordinary sense (for example, they are the parts defined by  $G_U$  in Example 43). We shall gain some insight into the structure of parts of a formula. Show the following. *Let  $\vec{x}$  be a formula and  $\vec{y}$  be a substring that is a formula. Then there is an index  $\vec{z}$  such that  $\vec{y}\vec{z}$  is a subformula of  $\vec{x}$ .* Thus, any context free grammar that generates the set of formulae proceeds basically like  $G_U$  modulo appending some index at the end of a formula.

**Exercise 38.** Use the previous exercise to show that there is no strongly independent context free grammar avoiding syncategorematic rules for  $\text{UBool}^\S$ .

**Exercise 39.** Let  $L$  be a language with finite expressive power (that is, with  $\mu[L]$  finite). Then if  $L$  is weakly  $\mathcal{C}$ , it is strongly  $\mathcal{C}$ . Give an example of a language that is weakly  $\mathcal{C}$  but not superstrongly  $\mathcal{C}$ . *Remark.* For the proof to go through we need some trivial assumptions on  $\mathcal{C}$ . I propose to assume that membership in  $\mathcal{C}$  depends only on the fact that all  $\mathcal{J}(f)$  have a certain property  $\mathcal{P}$ .

### 3.6 Indeterminacy in Interpreted Grammars

This section is largely based on [Kracht, 2007]. We have considered in Section 2.4 the notion of an indeterminate grammar. I shall now pick up that theme again, fulfilling my earlier promise to show that if we are serious about compositionality then indeterminacy is not an option.

**Definition 3.19** *Let  $E$  and  $M$  be sets of exponents and meanings, respectively. An **indeterminate interpreted grammar over  $E \times M$**  is a pair  $\langle \Omega, \mathcal{J} \rangle$ , where  $\Omega$  is a signature and for every  $f \in F$ ,  $\mathcal{J}(f) \subseteq (E \times M)^{\Omega(f)+1}$ . The **language** generated by  $G$ , in symbols  $L(G)$ , is defined to be the least set  $S$  such that for every  $f \in F$  and all  $\sigma_i \in E \times M$ ,  $i < \Omega(f)$ , and  $\tau \in E \times M$ :*

$$(3.114) \quad \text{if for all } i < \Omega(f) : \sigma_i \in S \text{ and if } \langle \sigma_0, \dots, \sigma_{\Omega(f)-1}, \tau \rangle \in \mathcal{J}(f) \text{ then } \tau \in S$$

This is the broadest notion, allowing to form signs from signs.  $G$  is **autonomous** if the exponent of the output sign is independent of the meanings. We can explicate this as follows. For every  $f$  and  $\sigma_i = \langle e_i, m_i \rangle$  and  $\sigma'_i = \langle e_i, m'_i \rangle \in E \times M$  (where  $i < \Omega(f) + 1$ ) such that  $m_{\Omega(f)} = m'_{\Omega(f)}$ :

$$(3.115) \quad \text{If } \vec{\sigma} \in \mathcal{J}(f) \text{ then } \vec{\sigma}' \in \mathcal{J}(f)$$

This can be restricted to the language generated by the grammar, but we refrain from introducing too many fine distinctions. Similarly,  $G$  is **compositional** if for every  $f$  and  $\sigma_i = \langle e_i, m_i \rangle$  and  $\sigma'_i = \langle e'_i, m_i \rangle \in E \times M$  (where  $i < \Omega(f) + 1$ ) such that  $e_{\Omega(f)} = e'_{\Omega(f)}$ :

$$(3.116) \quad \text{If } \vec{\sigma} \in \mathcal{J}(f) \text{ then } \vec{\sigma}' \in \mathcal{J}(f)$$

Let us draw some consequences. If  $G$  is indeterminate, we say that the indeterminacy of  $G$  is **semantically spurious** if for all  $\sigma_i \in L(G)$ ,  $i < \Omega(f) + 1$ , if

$\langle \sigma_0, \dots, \sigma_{\Omega(f)-1}, \langle e, m \rangle \rangle \in \mathcal{J}(f)$  and  $\langle \sigma_0, \dots, \sigma_{\Omega(f)-1}, \langle e, m' \rangle \rangle \in \mathcal{J}(f)$  then  $m = m'$ . This means that  $G$  restricted to its own language actually has a semantically functional equivalent (the exponents may still be indeterminate even inside the language). *Syntactically spurious indeterminacy* would be defined dually.

**Proposition 3.20** *Let  $L$  be unambiguous and assume that  $G$  is an indeterminate interpreted grammar for  $L$ . Then the indeterminacy of  $G$  is semantically spurious.*

The proof is straightforward. If we generate two signs  $\langle e, m \rangle$  and  $\langle e, m' \rangle$  from the same input (in fact from any input), then  $m = m'$ .

Thus,  $G$  is already autonomous (at least extensionally). For an unambiguous grammar it may still be possible to write an indeterminate compositional (and hence independent) grammar. In the remainder of this section we study boolean logic and give both a positive and a negative example. Recall from Example 22 boolean logic in Polish Notation and the unbracketed notation as given in Example 43. Here we shall give yet another formulation, this time with obligatory bracketing. The details are similar to those in Example 43. The only difference is that the alphabet also contains the symbols  $/$  ( $/$  and  $/$ ) and that the formation rules insert these brackets every time a new constituent is being formed:

$$\begin{aligned}
 \mathcal{J}(f_0)() &:= \langle \mathbf{p}, [\varepsilon] \rangle \\
 \mathcal{J}(f_1)(\langle \vec{x}, U \rangle) &:= \langle \vec{x} \mathbf{0}, [\dagger(U) \mathbf{0}] \rangle \\
 \mathcal{J}(f_2)(\langle \vec{x}, U \rangle) &:= \langle \vec{x} \mathbf{1}, [\dagger(U) \mathbf{1}] \rangle \\
 \mathcal{J}(f_3)(\langle \vec{x}, U \rangle) &:= \langle (\neg \vec{x}), \text{Val} - U \rangle \\
 \mathcal{J}(f_4)(\langle \vec{x}, U \rangle, \langle \vec{y}, V \rangle) &:= \langle (\neg \vec{x} \wedge \vec{y}), V \cap U \rangle \\
 \mathcal{J}(f_5)(\langle \vec{x}, U \rangle, \langle \vec{y}, V \rangle) &:= \langle (\neg \vec{x} \vee \vec{y}), V \cup U \rangle
 \end{aligned}
 \tag{3.117}$$

We call this language Bool. This grammar defines the semantics of a formula to be a set of valuations. There is a different semantics, which is based on a particular valuation  $\beta$ , and which is defined as follows.

$$\beta(\varphi) = \begin{cases} 1 & \text{if } \beta \in [\varphi] \\ 0 & \text{else.} \end{cases}
 \tag{3.118}$$

**Example 44.** Let  $B$  be the string language of boolean expressions. Pick a valuation  $\beta$  and let

$$L := \{ \langle \varphi, \beta(\varphi) \rangle : \varphi \in B \}
 \tag{3.119}$$

Consider an indeterminate string grammar  $G = \langle F, \Omega \rangle$  for it, for example the grammar from Exercise 22. Put  $F_2 := \{f^0, f^1 : f \in F\}$  and let  $\Omega^2(f^0) := \Omega^2(f^1) := \Omega(f)$ . Finally, put

$$(3.120) \quad \begin{aligned} \mathcal{J}(f^0) &:= \{ \langle \langle e_i, m_i \rangle : i < \Omega(f) + 1 \rangle : \langle e_i : i < \Omega(f) + 1 \rangle \in \mathcal{J}(f), \\ &\quad \beta(e_{\Omega(f)}) = 0, m_{\Omega(f)} = 0 \} \\ \mathcal{J}(f^1) &:= \{ \langle \langle e_i, m_i \rangle : i < \Omega(f) + 1 \rangle : \langle e_i : i < \Omega(f) + 1 \rangle \in \mathcal{J}(f), \\ &\quad \beta(e_{\Omega(f)}) = 1, m_{\Omega(f)} = 1 \} \end{aligned}$$

So the relations are split into two, where the first set contains the tuples whose last member is a formula that is true under the valuation, and the second relation collects the other tuples. This is an indeterminate interpreted grammar. Call it  $G^2$ . It might be that the newly created symbols are actually interpreted by functions, but this does not have to be the case. A case in point is Example 22, the grammar for Polish Notation. A given string of length  $n$  may possess up to  $n$  adjunction sites, thus making the resulting grammar  $G^2$  indeterminate again. Consider for example the string  $/\wedge p \wedge p \wedge p p/$ . Assume that  $\beta(p) = 1$ . Then the value of that formula is also 1. The string  $/\wedge p/$  can be adjoined at several places, marked here with  $\circ$ :

$$(3.121) \quad \circ \wedge p \circ \wedge p \circ \wedge p \circ p$$

In all cases the resulting formula has value 1, but it is clear that we do not even need to know this. There are more than two output strings, so some of them must have the same truth value.  $\otimes$

That the semantics is finite is used essentially in the proof. The example is of course quite dissatisfying; the functions are undefined depending on what the meaning of the string is. On the other hand, there may be a way to circumvent the dependency on semantics, that is to say, the fact that the meaning figures in the definition of the functions may just be an artefact of the way we defined them. However, there are different examples to show that indeterminacy is not such a good idea.

In what is described below I shall look into the possibility of defining a compositional adjunction grammar for the language of boolean expressions, where  $\varphi$  has as its meaning the set of all assignments that make it true. The rest of this section is devoted to the proof of the following theorem.

**Theorem 3.21** *There is no independent tree adjunction grammar (and hence no compositional tree adjunction grammar) for Bool in which all meaning functions are total.*

Independence is of course essential. Since Bool is a function, there can also be no compositional grammar, for autonomy can be guaranteed at no cost: whatever dependency the exponents show on the meanings can be expressed as a dependency on the exponent, since we can guess the meaning from the exponent. The argument works as follows. We use only formulae that contain conjunction. Consider a tree that has  $n$  distinct occurrences of the same subtree  $U$  into which one can adjoin  $T$  (recall that we are doing tree adjunction now). The adjunction can be performed  $n$  times, however the first and the last are distinct. In the first we replace  $U$  by some subtree  $U'$ , which may be a new subtree. In the last round we make the last occurrence of  $U$  disappear (unless of course  $U'$  contains an instance of  $U$ ). A case in point are subtrees representing binary sequences (which are part of variables). Adjunction changes the name of the variable. Thus one and the same adjunction must sometimes be paired with the identity map and sometimes result in a change (when the variable to which we used to adjoin disappears). This is a contradiction. However, since the same adjunction tree may be paired with any number of semantic maps we are on the safe side if we can produce an arbitrary amount of occurrences of the same variable. This is done as follows. We show that there is a tree  $T$  such that

- ① The root of  $T$  can be adjoined to.
- ②  $T$  introduces an entire formula to which adjunction is licit.

If such a tree does not exist, it is because for any tree  $T'$  either (a) the root of  $T'$  cannot be adjoined to, or (b) it fails to introduce an entire formula, or (c) any entire formula it introduces cannot be adjoined to. A formula  $\vec{x}$  is called  $n$ -homogeneous of order  $p$  if either (1)  $n = 0$  and  $\vec{x}$  is a variable of length  $> p$ , or (2)  $n > 0$  and  $\vec{x} = (\vec{y} \wedge \vec{z})$  where both  $\vec{y}$  and  $\vec{z}$  are  $n - 1$ -homogeneous of order  $p$ . We shall show

For any given tree adjunction grammar  $G$  for  $\text{Bool}^\wedge$  there are  $n$  and  $p$  such that  $G$  any  $n$ -homogeneous formulae of order  $p$  must be generated in  $G$  using a tree satisfying ① and ②.

This will yield the required proof. Notice that since  $\text{Bool}^\wedge$  is a fragment the result extends to  $\text{Bool}$ .

Before we can embark on the proof, we have to make some preparations.

**Definition 3.22** *Let  $L \subseteq E \times M$  be an interpreted language and  $D \subseteq E$ . Then  $L \upharpoonright D := L \cap (D \times M)$  is the  $D$ -**fragment** of  $L$ . If  $E = A^*$  and  $D = B^*$  then we also write  $L \upharpoonright B$  in place of  $L \upharpoonright B^*$ .*

The case where we restrict to a subalphabet is the one which we shall use here. We shall study the following fragments of  $\text{Bool}$ :

$$\begin{aligned} \text{Var} &:= \text{Bool} \upharpoonright \{\mathbf{p}, \mathbf{0}, \mathbf{1}\} \\ (3.122) \quad \text{Bool}^\wedge &:= \text{Bool} \upharpoonright \{(\ , \ ), \mathbf{0}, \mathbf{1}, \mathbf{p}, \wedge\} \\ \text{Bool}^\neg &:= \text{Bool} \upharpoonright \{(\ , \ ), \mathbf{0}, \mathbf{1}, \mathbf{p}, \neg\} \end{aligned}$$

Now assume  $G$  is a grammar for  $L$ . Then for every  $f$ , let

$$(3.123) \quad \begin{aligned} f^\varepsilon \upharpoonright D &:= f^\varepsilon \upharpoonright (D \times M) \\ f^\mu \upharpoonright D &:= f^\mu \upharpoonright (D \times M) \end{aligned}$$

Finally,

$$(3.124) \quad f \upharpoonright D := (f^\varepsilon \upharpoonright D) \times (f^\mu \upharpoonright D)$$

For this to be well defined we need to show that the functions stay inside  $D \times M$ . For a string  $\vec{x}$  and a symbol  $a$ , let  $\#_a(\vec{x})$  denote the number of occurrences of  $a$  in  $\vec{x}$ . For  $E = A^*$ ,  $f : E^n \rightarrow E$  is **pseudoadditive** if for every  $a \in A$ : either  $\#_a(\vec{x}_i) = 0$  for all  $i < n$  and then  $\#_a(f(\vec{x}_0, \dots, \vec{x}_{n-1})) = 0$  or

$$(3.125) \quad \#_a(f(\vec{x}_0, \vec{x}_1, \dots, \vec{x}_{n-1})) \geq \sum_{i < n} \#_a(\vec{x}_i)$$

If equality holds,  $f$  is called **additive**. A grammar is additive if every function is. (A combination of Structure Preservation and Syncategorematicity Prohibition guarantees additivity, actually.) Now suppose further that our grammar is additive and that  $D = B^*$ . Then if all the  $\vec{x}_i$  are in  $B^*$ , so is  $f^\varepsilon(\vec{x}_0, \dots, \vec{x}_{n-1})$ . Hence we have a grammar

$$(3.126) \quad \begin{aligned} (\mathcal{J} \upharpoonright B)(f) &:= \mathcal{J}(f) \upharpoonright B \\ G \upharpoonright B &:= \langle \Omega, \mathcal{J} \upharpoonright B \rangle \end{aligned}$$


Now,  $G \dot{\vdash} B$  generates a subset of  $L$ , by construction. Moreover, by induction on the term  $t$  we can show that if  $\iota_G(t) \in (B^* \times M)$  then  $\iota_{G \dot{\vdash} B}(t) = \iota_G(t)$ . It follows that  $G \dot{\vdash} B$  generates *exactly*  $G \dot{\vdash} B$ .

**Proposition 3.23** *Suppose that  $G$  is an additive compositional grammar for  $L$ . Then  $G \dot{\vdash} B$  is an additive compositional grammar for  $L \dot{\vdash} B$ .*

Thus if  $G$  is an adjunction grammar so is  $G \dot{\vdash} B$ .

**Example 45.** We look in some detail at the fragment Var. Syntactically, we may generate this language by admitting adjunction anywhere except before the letter /p/. Yet, for every weakly compositional grammar  $G$  there can only be a bounded number of adjunction sites for most variables. Consider, for example, the adjunction string  $\langle 1, \varepsilon \rangle$  and the variable

$$(3.127) \quad p000000 \cdots 0$$

For simplicity we fix the adjunction sites to be of the form  $\langle p\vec{x}, \vec{y}, \varepsilon \rangle$ . Depending on  $\vec{x}$  we get a different variable. Thus, for any given rule only one of the adjunction sites from  $\{\langle p0^m, 0^{k-m}, \varepsilon \rangle : m \leq k\}$  may be chosen for the rule. One way to achieve this is to only use adjunction strings of the form  $\langle \vec{x}, \varepsilon \rangle$  and adjunction sites of the form  $\langle p, \vec{y}, \varepsilon \rangle$ . 

**Example 46.** Another place where caution needs to be exercised when doing adjunction is the following. Let  $\varphi$  be a formula consisting of variables and their negations. Suppose that  $\varphi$  contains a variable and its negation, as in

$$(3.128) \quad (p01 \wedge (\neg p01))$$

Then no valuation satisfies  $\varphi$ . In other words, we have  $\langle \varphi, \emptyset \rangle \in \text{Bool}$ . Consider now what happens if we adjoin to one of them some string. Then one of the occurrences disappears and the formula may suddenly have valuations that satisfy it. Let us adjoin 1, for example:

$$(3.129) \quad (p101 \wedge (\neg p01))$$

Any valuation mapping /p101/ to 1 and /p01/ to 0 satisfies this formula. Suppose that  $G$  is compositional. (Weakness does not add anything interesting here.) As  $G$

has only boundedly many rules, there can only be boundedly many values computed from any given meaning. Thus, if  $G$  has  $k$  rules,  $\text{card}(\{f^\mu(\emptyset) : f \in G\}) \leq k$ . It follows that adjunction can target only a restricted set of contradicting variables.

⊛

Adjoining binary strings to variable names is a good case to show that the independence of syntax and semantics is actually useless for practical applications. In the case of adjoining other strings, we shall see that their adjunction is actually syntactically restricted. Let us look in some detail at the possible adjunction strings for the language. We concentrate on  $\text{Bool}^\wedge$ . It is not difficult to see that in a string for  $\text{Bool}^\wedge$

$$(3.130) \quad \#_c(\vec{x}) = \#_j(\vec{x}) = \#_\wedge(\vec{x}) = \#_p(\vec{x}) - 1$$

Thus, for an adjunction string  $\langle \vec{x}, \vec{y} \rangle$  we must have

$$(3.131) \quad \#_c(\vec{x}\vec{y}) = \#_j(\vec{x}\vec{y}) = \#_\wedge(\vec{x}\vec{y}) = \#_p(\vec{x}\vec{y})$$

The number of /0/ and /1/ by contrast is unconstrained. For a given string call  $\#_c(\vec{x}) - \#_j(\vec{x})$  the **balance** of  $\vec{x}$ . Let us say that a  $\vec{x}$  is **semibalanced** if the balance of  $\vec{x}$  is zero and the balance of every prefix of  $\vec{x}$  is nonnegative. An adjunction string  $\langle \vec{x}, \vec{y} \rangle$  is **semibalanced** if  $\vec{x}\vec{y}$  is. Consider an occurrence of a string  $\vec{z}$  in  $\vec{x}$ . It can be given as a pair  $\langle \vec{u}, \vec{w} \rangle$  such that  $\vec{u}\vec{z}\vec{w} = \vec{x}$ . This occurrence has **degree of embedding**  $k$  if the balance of  $\vec{u}$  is  $k$ . Note some facts on formulae which are easily established by induction.

- ① Every formula is semibalanced.
- ② Every formula is either a variable or contains exactly one function symbol with degree of embedding 1.
- ③ If  $\vec{x}$  is a formula and  $\vec{y}$  a semibalanced substring that begins with an opening bracket then  $\vec{y}$  is a formula.
- ④ If  $\vec{x}$  is a formula and  $\vec{y}$  a substring that is a formula, then replacing an occurrence of  $\vec{y}$  in  $\vec{x}$  by a formula yields a formula again. (In other words, the language is transparent.)

If  $\vec{x}$  is not a variable we call the unique symbol occurrence of degree 1 the **main symbol** of  $\vec{x}$ .



**Lemma 3.24** *If both  $\vec{u}\vec{v}$  and  $\vec{u}\vec{x}\vec{v}$  are formulae then  $\vec{x}$  is a binary string.*

Here is now a central theorem, which I shall not prove here and instead refer to [Kracht, 2007].

**Lemma 3.25** *An adjunction string for  $\text{Bool}^\wedge$  must be semibalanced.*

We now analyse in some depth the possible adjunction strings. Notice that the adjunction strings must be semibalanced. Furthermore, we can restrict our attention to cases in which just one occurrence of each bracket is introduced. For either (i) one part of the adjunction string contains a sequence  $/(\cdots)/$ , and so a formula. Or (ii) the adjunction string has the form  $\langle \cdots (\cdots (\cdots, \cdots) \cdots) \cdots \rangle$ . In Case (i), we first introduce the string without the subformula (just introducing  $/p/$ ) and require adjunction (by choosing an appropriate label) that will introduce the remainder. In Case (ii) we pretty much do the same: the innermost brackets enclose a formula, and so do the outermost brackets. We can perform the same adjunction in two stages.

Assume that this is so. This means that in underived adjunction strings,  $/(/$  is in the left part,  $/)/$  in the right part. We distinguish two subcases: (I)  $/\wedge/$  is in the left part, (II)  $/\wedge/$  is in the right part.

(Case I).  $/\wedge/$  is in the left part. Then the adjunction pair looks like this (ignoring  $/0/$ ,  $/1/$  and  $/p/$ ):

$$(3.132) \quad \langle \cdots \wedge \cdots (\cdots, \cdots) \cdots \rangle$$

There can in fact be no symbol between  $/\wedge/$  and  $/(/$  as no binary string can occur there. Furthermore, after a closing bracket a formula can contain only  $/\wedge/$  or  $/)/$ . This reduces the adjunction string to the following form:

$$(3.133) \quad \langle \cdots \wedge (\cdots, \cdots) \rangle$$

Since neither part of the adjunction string contains a complete formula,  $/p/$  must occur right after  $/(/$ . (3.134) is the final form, where dots represent some binary string.

$$(3.134) \quad \langle \cdots \wedge (p \cdots, \cdots) \rangle$$

(Case II).  $/\wedge/$  is in the right part. Since neither part contains an entire formula,  $/\wedge/$  does not precede  $/)/$ .

$$(3.135) \quad \langle \dots (\dots, \dots) \dots \wedge \dots \rangle$$

$/(/$  can only be preceded by  $/(/$  and  $/\wedge/$  in a formula; thus  $/(/$  is not preceded by anything. Also,  $/\wedge/$  must follow  $/)/$  immediately. So, we are left with the following choices, with dots again representing binary strings:

$$(3.136) \quad \langle (\mathbf{p} \dots, \dots) \wedge \dots \rangle \qquad \langle (\dots, \dots) \wedge \mathbf{p} \dots \rangle$$

$/\wedge/$  is not followed by a binary symbol, so we get

$$(3.137) \quad (\alpha) \quad \langle (\mathbf{p} \dots, \dots) \wedge \rangle \qquad (\beta) \quad \langle (\dots, \dots) \wedge \mathbf{p} \dots \rangle$$

The type  $(\alpha)$  is ruled out since it can nowhere be entered into a formula. To see this, let  $\langle \vec{u}, \vec{v}, \vec{w} \rangle$  be the context. The result of adjunction is

$$(3.138) \quad \vec{u}(\underline{\mathbf{p} \dots \vec{v} \dots}) \wedge \vec{w}$$

What the brackets enclose in (3.138) is a formula. Also,  $\vec{w}$  has a prefix  $\vec{x}$  that is a formula.  $\vec{v}$  may not begin with  $/\dots)/$  nor with  $/\dots(/$ , so it begins with  $/\dots \wedge/$ . And it ends likewise in  $/\mathbf{p} \dots/$  or in  $/)/$ . Since the  $\vec{u}\vec{v}\vec{w}$  is a formula and contain the sequence  $\vec{v}\vec{x}$ ,  $\vec{x}$  cannot begin with a bracket, and so must be a variable. And  $\vec{v}$  cannot end in  $/)/$ , neither can it end in  $/(\dots/$ . Contradiction. Thus only the Type  $(\beta)$  needs to be considered. In this type,  $/(/$  cannot be followed by a binary string. So we are down to the case (3.139).

$$(3.139) \quad \langle (\dots) \wedge \mathbf{p} \dots \rangle$$

In Case (I) there is an occurrence of  $/\wedge/$  that is not immediately preceded by a closing bracket, and in Case (II) there is an occurrence of  $/\wedge/$  that is not followed by an occurrence of an opening bracket.

Now that we know about their identity, let us check where these adjunction strings can be inserted. In both cases it is easy to see that the kernel must contain an occurrence of  $/\wedge/$ . The occurrence of  $/\mathbf{p}/$  in (3.139) cannot be separated by an opening bracket from its preceding  $/\wedge/$ . For if we did this, we would have to insert a closing bracket at some later position. This can be only after the adjunction string (3.139), thus forming crossing adjunction sites. Contradiction. Now let us

look at (3.134). Suppose that the left part begins with a proper binary string. Then  $/\wedge/$  is preceded by some variable, and we can by the same argument not insert a closing bracket. Thus, we are left with only the following case.

$$(3.140) \quad \langle \wedge(p \cdots, \cdots) \rangle$$

Let us now see how we can derive an  $n$ -homogeneous string of order  $p$ ,  $p$  large enough so that no trees of Type (c) can be used. The left periphery of this string consists in  $n$  opening brackets, which cannot be derived using rules of the form (3.140). Thus Type (b) with  $/\wedge/$  in the left part (Case I) is ruled out. Type (b) (Case II) leads to (3.139) as the only possible adjunction string. But that cannot be used either, since there must be an opening bracket between the occurrence of  $/\wedge/$  and  $/p/$ , since the formula is homogeneous. So, only trees of Type (a) can be used at the left periphery. They have this form.

$$(3.141) \quad (*) \quad \langle (\vec{x}\wedge, ) \rangle \quad (\dagger) \quad \langle (, \wedge\vec{x}) \rangle$$

Here  $\vec{x}$  must be a formula. Make  $n > pq$  where  $q$  is the number of nonterminals. Then it is the Type  $(\dagger)$  that we need for the left periphery since  $\vec{x}$  contains less than  $p$  symbols (and is not composed entirely of brackets). Now, if we have  $n$  opening brackets, there are  $q$  adjunction sites stacked for  $(\dagger)$  inside each other. One pair of them has the same nonterminal, and from this we can get a (possibly derived) adjunction tree satisfying ① and ②.

**Example 47.** I give a letter by letter translation of Bool into English:

$$(3.142) \quad \begin{aligned} t(p) &= /Jack \text{ sees a boy}/ \\ t(C) &= \varepsilon \\ t() &= \varepsilon \\ t(\emptyset) &= /who \text{ sees a girl}/ \\ t(1) &= /who \text{ sees a boy}/ \\ t(\wedge) &= /who \text{ sees no one and}/ \\ t(\vee) &= /who \text{ sees no one or}/ \\ t(\neg) &= /it \text{ is not the case that}/ \end{aligned}$$

Now define

$$(3.143) \quad \begin{aligned} s(\varepsilon) &:= /who \text{ sees no one.}/ \\ s(a \wedge \vec{x}) &:= t(a) \wedge s(\vec{x}) \end{aligned}$$

This gives us, for example,

$$(3.144) \quad s((p\mathbf{0} \wedge (\neg p))) = / \text{Jack sees a boy who sees a girl who sees} \\ \text{no one and it is not the case that} \\ \text{Jack sees a boy who sees no one.} /$$

Consider the set  $B = \{j\} \cup \{b\vec{x} : \vec{x} \in (\mathbf{0} \mid \mathbf{1})^*\} \cup \{g\vec{x} : \vec{x} \in (\mathbf{0} \mid \mathbf{1})^*\}$ . Here  $j$  is Jack,  $b\vec{x}$  is the boy number  $\vec{x}$  and  $g\vec{x}$  the girl number  $\vec{x}$ . Let  $U \subseteq (\mathbf{0} \mid \mathbf{1})^*$ . Define  $R(U)$  as follows.

$$(3.145) \quad R(U) := \begin{cases} \{ \langle b\mathbf{0}\vec{x}, g\vec{x} \rangle : \vec{x} \in (\mathbf{0} \mid \mathbf{1})^* \} \\ \cup \{ \langle g\mathbf{0}\vec{x}, b\vec{x} \rangle : \vec{x} \in (\mathbf{0} \mid \mathbf{1})^* \} \\ \cup \{ \langle b\mathbf{1}\vec{x}, b\vec{x} \rangle : \vec{x} \in (\mathbf{0} \mid \mathbf{1})^* \} \\ \cup \{ \langle g\mathbf{1}\vec{x}, b\vec{x} \rangle : \vec{x} \in (\mathbf{0} \mid \mathbf{1})^* \} \\ \cup \{ \langle j, b\vec{x} \rangle : \vec{x} \in U \} \end{cases}$$

What can be shown is that the translation of  $/p\vec{x}/$  is true in  $\langle B, j, R(U) \rangle$  (with  $R(U)$  interpreting the relation of seeing and  $j$  interprets the constant “Jack”) iff  $\vec{x} \in U$ . Thus we have a translation into English that preserves synonymy. Though the argument is not complete (for the reason that the English examples do away with brackets and so introduce ambiguity), it does serve to transfer Theorem 3.21 to English.  $\odot$

**Exercise 40.** Show Lemma 3.24.

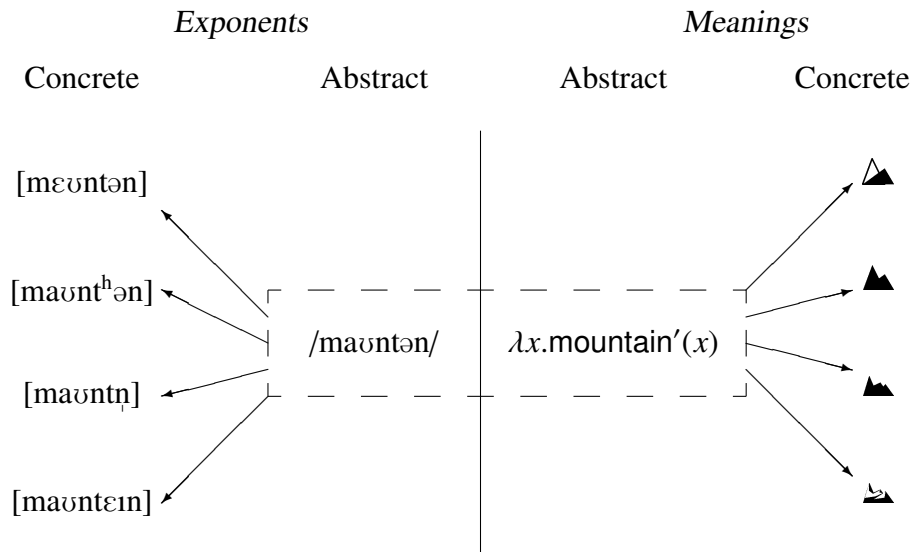
**Exercise 41.** Write a compositional adjunction grammar for Var.

**Exercise 42.** Let  $G$  be additive. Show that if  $\iota_G(t) \in (B^* \times M)$  then  $\iota_{G \circ B}(t) = \iota_G(t)$ .

### 3.7 Abstraction

At the end of this chapter I shall return to a problem that has been central in the development of modern linguistics: the definition of the *unit*. Units are *abstract* objects and are related to concrete things via *realisation*. As de Saussure already

Figure 3.6: Abstract Signs



insisted, the linguist almost always deals with abstract objects. The letter /b/, the sound [b], the genitive case—all these things are abstractions from observable reality. Thus, on the one hand the sign  $\langle /mountain/, \lambda x.mountain'(x) \rangle$  is the only thing that can be said to belong to *langage* as de Saussure defined it, but on the other hand it does not exist, unlike particular utterances of the word /mountain/ and particular mountains (the concept of mountainhood is an abstract object, the only thing we take to exist in the physical sense are individual mountains). An utterance of /mountain/ stands to the sequence of phonemes of /mountain/ in the same way as a particular mountain stands to  $\lambda x.mountain'(x)$ . In both cases the first is the concrete entity the second the abstract one, the one that is part of language. The picture in Figure 3.6 illustrates this. The main aim of this section is to give some mathematical background to the idea of abstracting units. Before I do so, I shall point out that there is no consensus as to how abstract language actually is. In earlier structuralism it was believed that only the abstract object was relevant. It was often suggested that only the contrast matters, and that the actual content of the contrasted items was irrelevant.

This view was applied to both phonology and semantics. It was thought that

there nothing matters to linguistics beyond the contrast, or feature, itself. It would then seem that the contrast between [p] and [b] could from the abstract viewpoint not be distinguished from the contrast between [p] and [t]; the labels “voicing” or “labial” are meaningless to phonology. Similarly, the meaning contrast between “short” and “tall” is formally indistinguishable from the contrast between “cold” and “hot”; all that can be said is that the contrasting items are different. This position—though not without merits, as we shall see—is nowadays not so popular. One reason among many is that it cannot explain how languages can change in a quasi continuous way and yet be underlyingly discrete. Additionally, it gives us no insight into why languages are the way they are, particularly when it comes to the certain bias that they display (for example to devoiced consonants in coda). Also, the precise content matters more often in language than structuralists were willing to admit. (The same predicament with respect to natural kinds and induction is discussed in [Gärdenfors, 2004].) The idea that we propose here is that the continuous change is the effect of a continuously changing surface realisation of abstract units. The contrasts are a matter of the underlying abstract language, and they get projected to the surface via realisation maps.

The picture that emerges is this. There are in total four domains:

1. concrete exponents (utterances)
2. abstract exponents (phonological representations)
3. concrete meanings (objects, things)
4. abstract meanings (semantics representations)

There are many-to-one maps from the concrete to the corresponding abstract domains. We take the pairing between concrete exponents and concrete meanings as given; this is the data. The pairing between abstract exponents and abstract meanings is postulated and likewise the correspondence concrete-to-abstract. In this scenario it becomes clear why we can have on the one hand agreement about the extensional language, say English, and yet disagreement about what the nature of representations is. Moreover, it becomes clear why it is that different people possess the same language yet possess a different grammar.

We take the notion of (concrete) language in the purely extensional sense: a set of pairs between utterances and concrete relata. For concreteness, we shall

just assume the *relata* to be things. Thus let us be given a set  $U$  of **utterances** and a set  $R$  of (physical) **relata**, that is, objects of the world. Language in the extensional sense is a subset of  $U \times R$ . A pair  $\langle u, r \rangle$  is in  $L$  if and only if  $u$  means  $r$  in that language. Thus, if a particular house  $h$  can be referred to by a particular utterance  $h'$  of /house/, then  $\langle h', h \rangle$  is a member of English. Some people may worry that  $R$  is potentially too big (something like the universal class), but from a methodological point of view nothing much is lost if we suitably restrict  $R$ . (In set theory one usually considers models of bounded size, the bound being suitably high. In a subsequent step one looks at the dependency of the result of the size of the bound.)

Both sets  $U$  and  $R$  are structured. The intrinsic structure of  $R$  is much harder to establish, so we just look at  $U$ . To simplify matters again, we assume that  $U$  consists of occurrences of sound bits (but see [Scollon and Wong Scollon, 2003] for an eloquent argument why this is wrong). Then we may be justified in assuming that only the intrinsic physical quality really matters, in other words: we can shift  $u$  in time (and place) without affecting its signalling potential. Thus, from now on we deal not with actual utterances but with what we call “sound bits”. Sound bits are what you store in a file on a computer to play it to someone (or yourself) any time you want. This is nowadays used a lot in talking machines (like GPS systems, dialog systems or elevators). Now let  $\odot$  be the append operation on sound bits. Such an operation can easily be realised on a computer, and this technique is also widely used in technical applications.  $\odot$  restricted to  $U$  becomes a partial operation. This is because there are phonotactic restrictions on the combinations of sounds. Given this operation  $\odot$  it is possible to segment sound bits into smaller units. In this way an utterance of /house/ can be segmented into a sequence of more primitive utterances, which are instances of some sound bits corresponding to the basic sounds of English. So, we propose a set  $P$  of primitive sound bits. The set  $P$  is an alphabet, and  $\odot$  the concatenation.  $P^*$  is the closure of  $P$  under  $\odot$ . Further,  $U$  is a subset of  $P^*$ .  $P$  is the set of **phones**. The choice of  $P$  is to some extent arbitrary; for example, in phonetics, an affricate is seen as a sequence of stop plus fricative (see for example [IPA, 1999]), but in phonology the affricates are often considered phonemes (= indecomposable). Similar problems are created by diphthongs. Although segmentation is a problem area, we shall not go into it here and instead move on to sketch the method of abstraction.

Both utterances and *relata* are concrete entities. My utterance  $u$  of /house/ at 11:59 today is certainly a concrete entity. We can record it and subsequently

analyse it to see if, for example, I really pronounced it in a proper English way or whether one can hear some German accent in it. Technically, each time you have the computer or tape recorder play  $u$  again you have a different utterance. Yet, we believe that *this* difference is merely temporal and that the relevant physical composition (pitch, loudness etc.) are all that is needed to make the two identical for the purpose of linguistics. That is to say, there is, hidden in the methodology at least, an underlying assumption that if  $u$  and  $u'$  are acoustically the same they are also linguistically the same. However, in our definitions we need not make any such assumption. If  $u$  cannot be reproduced since it is unique, so be it. If acoustic features really *are* sufficient this will actually be a result of the inquiry. Similarly, this building opposite of me is concrete; I can ask English speakers whether it qualifies to be called  $u$  (by playing them a copy of  $u$ ). Again there is a question whether calling this building a house today means that you will do so tomorrow; and if not why that is. If the difference in time is large enough (some decades) we cannot be sure that we are dealing with the same language again. If asking a different person we are not sure that s/he uses the words just like the one we asked before. And so on. Again, such difficulties do not affect so much the principles of the methodology described below they mainly delimit its factual applicability in concrete situations. However, once we know what the theoretical limitations of this methodology are—independently of its practical limitations—we can know better how to apply it.

The first tool in abstraction is the method of **oppositions**. We say that  $u$  and  $u'$  are **first degree  $L$ -equivalent**, in symbols,  $u \sim_L u'$ , if for all  $r \in R$ :  $\langle u, r \rangle \in L \Leftrightarrow \langle u', r \rangle \in L$ . Notice that this definition applies to entire utterances, and it tells us whether or not two particular utterances mean the same thing. Similarly, we say of two relata  $r$  and  $r'$  whether they are **first degree  $L$ -equivalent** if for all  $u \in U$ :  $\langle u, r \rangle \in L \Leftrightarrow \langle u, r' \rangle \in L$ . It is possible to factor out first-degree equivalence in the following way: let

$$(3.146) \quad [u]_1 := \{u' : u' \sim_L u\}, [r]_1 := \{r' : r' \sim_L r\}$$

Finally, put

$$(3.147) \quad L_1 := \{\langle [u]_1, [r]_1 \rangle : \langle u, r \rangle \in L\}$$

**Proposition 3.26** *Let  $u' \sim_L u$  and  $r' \sim_L r$ . Then  $\langle [u]_1, [r]_1 \rangle \in L_1$  if and only if  $\langle u', r' \rangle \in L$ .*



**Proof.** Assume that  $\langle [u]_1, [r]_1 \rangle \in L_1$ . Then  $\langle u, r \rangle \in L$ , by definition. Since  $u' \sim_L u$ , we also have  $\langle u', r \rangle \in L$ ; and since  $r' \sim_L r$  we have  $\langle u', r' \rangle \in L$ . This reasoning can be reversed.  $\square$

We can formalise this as follows.

**Definition 3.27** *Let  $U$  and  $R$  be sets,  $L \subseteq U \times R$  a language. Let  $f : U \rightarrow V$  and  $g : R \rightarrow S$  be maps such that the following holds:*

1. *If  $f(u) = f(u')$  then  $u \sim_L u'$ ;*
2. *If  $g(r) = g(r')$  then  $r \sim_L r'$ .*

*Then with  $L' := \{\langle f(u), g(r) \rangle : \langle u, r \rangle \in L\}$  the triple  $\langle f, g, L' \rangle$  is called an **abstraction of  $L$** .*

In particular, with the maps  $\varphi : u \mapsto [u]_1$  and  $\psi : r \mapsto [r]_1$  the triple  $\langle \varphi, \psi, L_1 \rangle$  is an abstraction of  $L$ . This is the maximal possible abstraction. Its disadvantage is that it is not ‘structural’. Consider a somewhat less aggressive compression which works as follows. Assume a representation of utterances as sequences of phones (so,  $U \subseteq P^*$  for some  $P$ ). Define  $p \approx_L p'$  if for all  $u \odot p \odot u'$ :

$$(3.148) \quad \text{If } u \odot p \odot u', u \odot p' \odot u' \in U \text{ then } u \odot p \odot u' \sim_L u \odot p' \odot u'$$

This can be phrased mathematically as follows:  $\approx_L$  is the largest weak congruence on  $\langle U, \odot \rangle$  which is contained in  $\sim_L$  (cf. Appendix A).

Standardly, the congruence  $\approx_L$  is used to define the **phonemes**. We say that  $p$  and  $p'$  are **allophones** of the same phoneme. Even though  $p$  and  $p'$  may not be exchangeable in every context, if they are, exchanging them causes no difference in meaning. In principle this method can also be applied to sequences of sounds (or strings), but that is only reluctantly done in phonology. One reason is that phonology likes the explanation for variability and equivalence to be phonetic: a combination of two sounds is ‘legal’ because it can easily be pronounced, illegal because its pronunciation is more difficult. Yet, with a different segmentation we can perform similar abstractions. Suppose we propose two units, say /good/ and /bett/, which occur in the gradation of the adjective ‘good’. In the positive we find /good/ while in the comparative we find /bett/. Thus, given that gradation proceeds by adding /Ø/ in the positive and /er/ in the comparative we can safely

propose that the two are equivalent. All it takes is to assume that only /good/ can be concatenated with /Ø/ and only /bett/ with /er/. There are two reasons why this is not a phonological but a morphological fact. The first is that there is no phonological law motivated by other facts that supports this equivalence. The other is that we can assign meaning to all the four parts; furthermore, we shall assume that /good/ and /bett/ have identical meaning, and with that the facts neatly fall out. One problem however remains in all these approaches: they posit *nonexistent parts*. To be exact: they are nonexistent as utterances in themselves; however, they do exist as parts of genuine utterances. This contradicts our earlier assumption that the set of valid forms of the language are only those that are first members of a pair  $\langle u, r \rangle$ . For now we accept forms that are not of this kind. Notice that the phonological abstraction did not require the units to be meaningful and proceeded just by comparing alternatives to a sound in context. The abstract units (phonemes) are not required to be in the language, nor are their parts. Thus the abstracted image  $L_1$  is of a new kind, it is a language (**langue**) in de Saussure's sense. It is certainly possible to do morphology along similar lines.

The language  $L$  can be identified with *parole*, while *langue* is  $L_1$ . However, we should be aware of the fact that while  $L$  is unique (given by experience),  $L_1$  is not. The most trivial way in which we can make a different abstraction is by using different abstract relata.

**Definition 3.28** Let  $\mathcal{A} = \langle \varphi, \psi, L_1 \rangle$  and  $\mathcal{B} = \langle \eta, \theta, L_2 \rangle$  be abstractions of  $L$ . We call  $\mathcal{A}$  and  $\mathcal{B}$  **equivalent** if

- ①  $\text{dom}(\varphi) = \text{dom}(\eta)$  and  $\text{dom}(\psi) = \text{dom}(\theta)$ ,
- ② there is a bijection  $i : L_1 \rightarrow L_2$  such that  $\eta \times \theta = i \circ (\varphi \times \psi)$

Put  $U = \text{dom}(\varphi)$  and  $R = \text{dom}(\psi)$ . Then we have the following situation.

$$(3.149) \quad \begin{array}{ccc} U \times R & \xrightarrow{\varphi \times \psi} & L_1 \\ & \searrow \eta \times \theta & \downarrow i \\ & & L_2 \end{array}$$

By definition there is an inverse map  $j : L_2 \rightarrow L_1$ . Finally, given a grammar  $G = \langle \Omega, \mathcal{J} \rangle$  for  $L = E \times M$  and an abstraction  $\mathcal{A} = \langle \varphi, \psi, L' \rangle$  we can define the abstracted

grammar  $G/\mathcal{A} := \langle \Omega, \mathcal{J}^{\mathcal{A}} \rangle$  for  $L'$  via  $\mathcal{A}$  as follows. For a sign  $\sigma = \langle e, m \rangle \in E \times M$  let  $\sigma^{\mathcal{A}} := \langle \varphi(e), \psi(m) \rangle$ , the abstraction of  $\sigma$ . Then for a function symbol define

$$(3.150) \quad \mathcal{J}^{\mathcal{A}}(f)(\sigma_0^{\mathcal{A}}, \dots, \sigma_{\Omega(f)-1}^{\mathcal{A}}) := (\mathcal{J}(f)(\sigma_0, \dots, \sigma_{\Omega(f)-1}))^{\mathcal{A}}$$

This is a familiar definition in mathematics; given an equivalence of elements we define the functions over the equivalence classes by picking representatives. This definition is sound only if the definition is actually independent of the choice of representatives. Otherwise the grammar becomes indeterminate.

**Example 48.** Here is an instructive example. Suppose

$$(3.151) \quad L = \{\langle \mathbf{a}, m \rangle, \langle \mathbf{b}, m \rangle, \langle \mathbf{c}, p \rangle, \langle \mathbf{ac}, n \rangle, \langle \mathbf{bc}, n' \rangle\}$$

The grammar consists of the following operations:

$$(3.152) \quad \begin{aligned} \mathcal{J}(f_0)() &:= \langle \mathbf{a}, m \rangle \\ \mathcal{J}(f_1)() &:= \langle \mathbf{b}, m \rangle \\ \mathcal{J}(f_2)() &:= \langle \mathbf{c}, p \rangle \\ \mathcal{J}(f_3)(\langle e, m \rangle, \langle e', m' \rangle) &:= \begin{cases} \langle \mathbf{ac}, n \rangle & \text{if } e = \mathbf{a}, e' = \mathbf{c} \\ \langle \mathbf{bc}, n' \rangle & \text{if } e = \mathbf{b}, e' = \mathbf{c} \\ \text{undefined} & \text{else} \end{cases} \end{aligned}$$

$/\mathbf{a}/$  and  $/\mathbf{b}/$  are  $L$ -equivalent. Put

$$(3.153) \quad L_1 = \{\langle \alpha, m \rangle, \langle \gamma, p \rangle, \langle \alpha\gamma, n \rangle, \langle \alpha\gamma, n' \rangle\}$$

Let  $\varphi : \mathbf{a}, \mathbf{b} \mapsto \alpha, \mathbf{c} \mapsto \gamma$  and  $1_M$  the identity on  $M = \{m, p, n, n'\}$ ; then  $\mathcal{A} := \langle \varphi, 1_M, L_1 \rangle$  is an abstraction. However, the grammar is not deterministic. Basically, the output of  $\mathcal{J}^{\mathcal{A}}(f_3)(\langle \alpha, m \rangle, \langle \gamma, p \rangle)$  must be both  $\langle \alpha\gamma, n \rangle$  and  $\langle \alpha\gamma, n' \rangle$ .  $\otimes$

It is important to note that the example does not show the impossibility of delivering a grammar. It just shows that the original grammar cannot necessarily be used as a canonical starting point. In general, (3.150) is a proper definition only if the congruence induced by  $\varphi$  and  $\psi$  is strong. Formally, the congruence induced by an abstraction is  $\theta_{\mathcal{A}}$ , where

$$(3.154) \quad \langle x, y \rangle \theta_{\mathcal{A}} \langle u, v \rangle \quad :\Leftrightarrow \quad \varphi(x) = \varphi(u) \text{ and } \psi(y) = \psi(v)$$

However, the condition is far too strong to be useful. A far more interesting case is when the congruence  $\theta_A$  is only weak. In that case the function is not independent of the choice of representatives; however, it is only weakly dependent. We will then say that  $\mathcal{J}^A(f)$  is simply the image of  $\mathcal{J}(f)$  under  $\varphi$  and  $\psi$ . Then in place of (3.150) we say that  $\mathcal{J}^A(\vec{\sigma})$  is defined if there are  $\tau_i$ ,  $i < \Omega(f)$ , such that  $\tau_i \theta_A \sigma_i$  for all  $i < \Omega(f)$  and  $\mathcal{J}(f)(\vec{\tau})$  is defined. And in that case

$$(3.155) \quad \mathcal{J}^A(f)(\sigma_0^A, \dots, \sigma_{\Omega(f)-1}^A) := (\mathcal{J}(f)(\tau_0, \dots, \tau_{\Omega(f)-1}))^A$$

Otherwise  $\mathcal{J}^A(\vec{\sigma})$  is undefined.


**Example 49.** There are two sounds in the phoneme /ɪ/, namely the voiced [ɪ] and the voiceless [ɪ̥]. They are mapped onto the same phoneme via  $\varphi$ . Now, in onset position, the combination [pɪ] does not exist in English, neither does the combination [bɪ̥]. Only the combination [pɪ̥] and the combination [bɪ] are possible. Consider the operation  $\odot$  of concatenation. [b]  $\odot$  [ɪ] is defined; [b]  $\odot$  [ɪ̥] is not. However,  $\varphi([ɪ]) = \varphi([ɪ̥])$ . Thus, congruences associated with the standard phonemicisation maps may be generally only weak congruences.  $\odot$

Likewise, a grammar for the abstracted language does not give rise to a grammar of the original language. In fact it may even be impossible to give one.

It is instructive to see that the combinatory restrictions on sounds do not necessarily determine a strong congruence. In fact, they rarely do. This has consequences worth pointing out. The most important concerns the standard definition of a phoneme. In the classical definition, two sounds are members of the same phoneme if they can be replaced for each other in any context without affecting meaning. It is clear that this must be read in the sense that replacing  $s$  for  $s'$  either yields a nonexistent form or else a form that has the same meaning. Otherwise, [ɪ] and [ɪ̥] might not be in the same phoneme for lack of intersubstitutability. However, that might not be enough to secure adequate phonemicisation. For it also turns out that the definition requiring the substitutability of *single* occurrences is also not enough if we have weak congruences.

**Example 50.** Let  $L := \{\langle aa, m \rangle, \langle bb, m \rangle\}$ . In this situation it seems justified to postulate a single phoneme  $\alpha$  with  $\varphi(a) = \varphi(b) = \alpha$ . The test which uses single substitutions indeed succeeds: we can replace /a/ by /b/ at any of the places, and

the result is either undefined or has the same meaning. The abstracted language is  $\{\langle \alpha\alpha, m \rangle\}$ .

Now look instead at the language  $L' := \{\langle aa, m \rangle, \langle bb, n \rangle\}$ . Here the definition based on single substitutions gives wrong results: if we change /a/ to /b/ once we get /ab/, which is *not* in the language. But if we change both occurrences we get /bb/, which however has different meaning. The abstracted language is the same. This cannot be correct. 

As the previous example showed, it is not enough to do a single replacement. It is not easy to come up with a sufficiently clear natural example. Vowel harmony could be a case in point. Recall that vowel harmony typically requires all vowels of a word to come from a particular set of vowels. In Finnish, for example, they may only be from  $\{\text{ä, e, i, ö, y}\}$  or from  $\{\text{a, e, i, o, u}\}$ . Consider now a bisyllabic word containing two occurrences of /ä/. Exchanging one of them by /a/ results in a nonharmonic string, which is therefore not a word. However, exchanging two or more occurrences may yield a proper word of Finnish. (Notice however that there are plenty of words that contain only one nonneutral vowel and so the logic of this argument is not perfect. For the latter kind of words may be enough to exclude those phonemicisations that are improper for the other words too.)



# Chapter 4

## Meanings

MEANINGS are the topic of this chapter. More precisely, it is abstract meanings that we want to characterise. 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 with exponents we must ask what meanings actually are and how they can be manipulated.

### 4.1 ‘Desyntactified’ Meanings

The present chapter is about what meanings *are*. Given the discussion of Section 3.7 we have two kinds of meanings to worry about: concrete meanings and abstract meanings. We shall for the most part consider a calculus of concrete meanings, but most of the results are actually independent of which of the two we study. Though much has been made of Putnam’s dictum that meanings (that is, concrete meanings) cannot be in a speaker’s head ([Putnam, 1975], see also [Gärdenfors, 2004]), the question whether or not that is so is actually peripheral to the question we are raising, namely, what meanings are and how they can be manipulated. For it threatens to focus the debate on questions of factual knowledge rather than matters of principle. Whether or not my concept of gold is the same as that of another person and which of us has the right concept is a question of factual detail. What matters in this book is what that concept of mine is and

how I use it; and similarly for any other person. Language is therefore subjective, I make no attempt at construction a language for a community of speakers. Communication is effected only via common expressions and must rely on intersubjective identity (or near identity) in their meaning.

We have said that meanings are given at the outset. It therefore seems to be of no relevance to ask what meanings are. However, there is a larger issue in the background that I cannot adequately treat in this book. The issue is that we cannot access concrete meanings as such; the only thing we can access is particular judgements. We have difficulties saying exactly what defines a book whereas we seem to be completely reliable in our judgement whether this or that thing is a book. And so there is a legitimate question as to whether the data we can access is the one we actually need.

While sentences are concrete since we can make them appear on tape or on paper, meanings are not directly observable. There is a long intellectual tradition to assume that meanings are structured (see [King, 2007] for a recent exposition). This position is adopted not only in philosophy but also in cognitive linguistics. Unfortunately, it is in practice hard to assess which particular structure the meaning of a given sentence has. In absence of a priori arguments the methodology should be to try to discover that structure from the given data. For it very often happens that our intuitions on meanings are obscured by our own language. What appears to be a semantic fact often enough is just a syntactic (or morphological) feature in disguise. In this way semantics is often infected with syntax. To counteract this trend I shall try to ‘desyntactify’ meanings. (See [Erdélyi Szabó *et al.*, 2007] for a more radical proposal of desyntactification.) In particular, below I shall identify some traits of semantic representations that I consider of purely syntactic nature: hierarchy, order, and multiplicity. Hierarchy shows up in the notion of a functional *type*; some meanings are functions, and therefore of a type that can take some (lower) types as arguments. This introduces an asymmetry into meanings that I claim does for the most part not exist in the meanings themselves. Order shows up in the notion of a *tuple*. Predicate logic explicates the meanings of formulae as relations, or sets of tuples. But where exactly the idea of a first member in a tuple or a second member is to be found in the actual denotation is unclear. Finally, although we can repeat a variable, we cannot repeat the same object. It follows that repetition may exist in syntax, but not in semantics. We shall look at these problem areas in more detail.

Frege is one of the proponents of the idea that there are “unsaturated” expres-



sions. For example, a function is unsaturated; it yields a value only when given an argument. The function  $x^2 + 5$ , in conventional notation, does not denote a number. We only get a number when we assign to  $x$  some value, say 3. Likewise, Frege argues, many words do not by themselves express a complete thought. They need certain argument places to be filled before this is the case. In this view, the phrase /Ate./ is unsaturated: it lacks a specification of the subject. Thus, only /John ate./ is complete. It is precisely this idea that has been exploited in Montague Grammar and Categorical Grammar. (Both of them diagnose this as a syntactic failure that is at root a type mismatch.) Unfortunately, it is unclear whether the incompleteness of /Ate./ is at all a semantic fact. There is an alternative line of analysis, which treats meanings as intrinsically complete (that is, propositional) and instead views the unacceptability of sentences such as /Ate./ as a purely syntactic fact of English. On this view, /Ate./ means “someone was eating something”. There are several reasons why this is a better idea for natural languages. The main one is that the correspondence between semantic arguments and syntactic positions is at best weak. The notion of eating involves both a subject and an object (and a time point, for that matter). An event of eating is constituted minimally by something being eaten and someone eating it. In order to pin down the exact meaning we need to know who ate what when. As it happens, /eat/ can also be used without an object. The standard approach (even in syntactic theory) has been to assume that in this case the sentence contains an empty object. Also, there are ways to convey the same meaning and yet use a fully grammatical construction, such as /There is eating./. What is or is not obligatorily expressed in a sentence varies greatly between languages. Some languages allow the subject to be dropped, for example. Finally, and relatedly, the analogy with the function is misleading in one important respect: while the argument to the function is a number, supplying a syntactic subject does not necessarily feed one. For should we assume that /John or Mary/ denotes a subject that we can feed to the verb, say in /John or Mary ate./? Similarly, /Someone ate./ contains a quantifier in subject position, something which is analysed not as an argument to the verb but rather as a functor. In my view, a syntactic argument serves to specify the identity of some object in question. This specification can be incomplete and thus the function once again lacks any specific value.

Montague has been impressed by the idea that syntactic requirements are at heart of semantic nature and has consequently endorsed the view that meanings are objects of a typed universe of functions. To implement this we may either choose a universe of the typed  $\lambda$ -calculus or some version of typed combinatory

logic. A type is a term of the language with a single binary symbol  $\rightarrow$  (you might want more type constructors, but that does not change the argument). There is a set of basic types, for example  $e$  and  $t$ , and one formation rule: If  $\alpha$  and  $\beta$  are types, so is  $\alpha \rightarrow \beta$ . Each type  $\alpha$  is associated with a set  $M_\alpha$  of denotations. It is generally required that  $M_\alpha \cap M_\beta = \emptyset$  whenever  $\alpha \neq \beta$ . This means that every object has at most one type. Furthermore, we require

$$(4.1) \quad M_{\alpha \rightarrow \beta} := (M_\beta)^{M_\alpha} := \{f : M_\alpha \rightarrow M_\beta\}$$

This leaves us with fixing only the sets  $M_b$  for basic  $b$ .

At its core Montague Grammar uses only two modes of combination: forward application and backward application.

$$(4.2) \quad \begin{aligned} A_>(\langle \vec{x}, m \rangle, \langle \vec{y}, n \rangle) &= \langle \vec{x} \smallfrown \vec{y}, m(n) \rangle \\ A_<(\langle \vec{x}, m \rangle, \langle \vec{y}, n \rangle) &= \langle \vec{x} \smallfrown \vec{y}, n(m) \rangle \end{aligned}$$

For  $A_>(\langle \vec{x}, m \rangle, \langle \vec{y}, n \rangle)$  to be defined  $m$  must be a function that can take  $n$  as its argument. This means that there are  $\alpha$  and  $\beta$  such that  $m$  is of type  $\alpha \rightarrow \beta$  and  $n$  of type  $\alpha$ . The result is then an object of type  $\beta$ .

Montague Grammar inherits from  $\lambda$ -calculus a number of traits; one is that functions cannot take their arguments simultaneously. A function can take only one argument at a time. This can be eliminated either by allowing simultaneous abstraction or by adding a pair constructor (as in the Lambek Calculus). However, linguists have supported the idea that functions take their arguments one by one. For this means that syntax is binary branching. This has been one of the central arguments in favour of Categorical Grammar. Thus, if we have a predicate with several arguments, we bring it into the desired form by ‘‘Currying’’, which is to abstract the arguments one by one. 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 idea of higher order types makes sense only if it is unequivocally clear what is argument and what is function. For if it is an intrinsic property of its

meaning of a verb is that it takes something as its argument there should be no doubt about that at all. Precisely this, however, has been a problematic issue for Montague Grammar. For on the one hand a singular proposition like “John is sick” is taken to be one where the verb denotes a semantic function taking the subject as its argument. On the other hand, quantified expressions have been argued to be structured in the opposite way: /everyone/ denotes a function in /Everyone is sick./. In order to avoid this mismatch, Montague decided to raise the denotation of /John/ so that it becomes a function over functions. But that was a technical manoeuvre. It was clearly not motivated from semantic considerations, but rather from syntactic uniformity. From here, it is a small step towards the type changing operations, which have been used extensively in [Landmann, 2004]. However, they threaten to undermine the idea that we have an intuitive grasp over the semantics of expressions.

Worse, it appears that the idea of the meaning of the syntactic subject as denoting the argument that is supplied to the function is generally unworkable. We can only say that it predicates *of* that argument. Modern semantics has basically adopted that latter view. However, if that is so, the whole function-argument asymmetry becomes arbitrary. And if we are free to view the subject alternatively as the argument to the verb or as the function I conclude that the distinction should be dropped altogether. Indeed, some philosophers and linguists have pursued a different semantics. One avenue is event semantics, which has been introduced to overcome not only the rigidity of the typing but also that of predicate logic itself (see [Parsons, 1994]).<sup>1</sup> Yet not everyone may be convinced. Therefore, to settle the matter we need empirical criteria. Additionally we need to see if there is a way to replace the typed universe with something else. For if there is not, then that in itself would weaken our position.

The preceding discussion can also be seen in a different light. Even if we grant that the meaning of /eat/ is a function there might be a question as to how that function is used in actual semantics. One camp holds that expressions are basically closed expressions. There are no free variables. One exponent of this view is P. Jacobson. The opposing view is that there is such a thing as free variables and there is no need to quantify them away. Proposals to this effect have been made in

---

<sup>1</sup>The need to free semantics from syntactic ‘impositions’ is also felt in Minimal Recursion Semantics ([Copestake *et al.*, 2005]). However, the latter is driven purely by concerns of practicability, and compensates for the lack of syntactic information by introducing labels. Such approaches, though widespread in computational linguistics do nothing to answer the questions that I have in mind here: namely whether semantics is independent of syntax.

[Kamp, 1981] and [Staudacher, 1987], among others. The disadvantage of closed expressions is that they make pronominal reference difficult (if not impossible). (But see [Jacobson, 1999; Jacobson, 2000; Jacobson, 2002] for an opposing view.)

As a consequence, DRT went the opposite way, namely not to abstract away arguments, but use formulae instead, with or without free variables. This however comes at a price. For if variables are no longer quantified away we must take proper care of them. 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 open propositions. An open proposition is a proposition that still needs certain variables to be filled. (These are exactly the “incomplete thoughts”.) Open propositions are the denotations of formulae. A formula is an expression of the form  $\varphi(x_0, x_1, \dots, x_{n-1})$  of type  $t$  ( $=$  truth value), where  $x_i$ ,  $i < n$ , are variables of any type. Thus, given an assignment of objects of appropriate type to the variables this expression will yield a truth value. A notable change to previous conceptions of truth, however, is that we consider an open proposition true exactly when it has a satisfying assignment. Thus, /eat/ becomes true exactly when someone is eating something at some moment. This is opposite to the standard conception in logic where an open proposition is considered true if there is no falsifying assignment; so /eat/ would be true if everyone eats everything at every moment. In our approach free variables are inherently existential, in standard predicate logic they are inherently universal. We should note that one problem that besets the free variable approach is that the choice of the actual variable inserted matters for the interpretation of the formula. However, it is patently clear that whether we use  $x_8$  or  $x_{11}$  is a matter of convenience.<sup>2</sup> Thus we have to devise a method to interpret such formulae and manipulate them in such a way that it does not make reference to the actual names of the variables. It is often thought that algebraic semantics has provided a solution to this problem, for example in the proposal by Quine. Here, meanings are relations, and there is no talk of variable names. Yet, now we need to talk about positions in a relation, which is not doing semantics either. We must namely also make explicit use of substitutions based on indices (see [Ben Shalom, 1996]). So this does not fully answer the complaint.

There is a well-known procedure to convert all meanings into open propositions. If  $m$  is a meaning of type  $\alpha$ ,  $\alpha \neq t$ , then replace it with  $x = m$ , where  $x$  is

---

<sup>2</sup>[Fine, 2007] has addressed this issue and came to the conclusion that meanings are relational. I will briefly discuss his proposal in 4.6.

of type  $\alpha$ . Consequently, signs of the form  $\langle \vec{x}, m \rangle$  are now replaced by signs of the form  $\langle \vec{x}, x = m \rangle$ . Now consider the rule of application:

$$(4.3) \quad A_{>}(\langle \vec{x}, m \rangle, \langle \vec{y}, n \rangle) = \langle \vec{x} \frown \vec{y}, m(n) \rangle$$

In the new semantics it becomes:

$$(4.4) \quad U_{>}(\langle \vec{x}, u = m \rangle, \langle \vec{y}, v = n \rangle) = \langle \vec{x} \frown \vec{y}, u = m \wedge v = n \wedge w = u(v) \rangle$$

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.

$$(4.5) \quad U_{>}(\langle \vec{x}, u = m(w) \rangle, \langle \vec{y}, v = n \rangle) = \langle \vec{x} \frown \vec{y}, u = m(w) \wedge v = n \wedge w = v \rangle$$

This rule simply conjoins the two meanings and unifies certain variables. The unification, by the way, is the semantic contribution of the rule itself, and cannot—on pain of reintroducing the same problematic meanings—be pushed into the meanings of the elements themselves. If  $m(w)$  is a function and has to be applied then we also have to feed to  $m(w)$  these additional arguments. In this way we can see to it that the generalised rule is as follows:

$$(4.6) \quad U_{>}^{ij}(\langle \vec{x}, \varphi(\vec{u}) \rangle, \langle \vec{y}, \chi(\vec{v}) \rangle) = \langle \vec{x} \frown \vec{y}, \varphi(\vec{u}) \wedge \chi(\vec{v}) \wedge u_i = v_j \rangle$$

Eliminating the equation we can alternatively write

$$(4.7) \quad U_{>}^{ij}(\langle \vec{x}, \varphi(\vec{x}) \rangle, \langle \vec{y}, \chi(\vec{y}) \rangle) = \langle \vec{x} \frown \vec{y}, \varphi(\vec{x}) \wedge [x_i/y_j]\chi(\vec{y}) \rangle$$

Thus we have the 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.

The standard rendering in predicate logic suffers from defects, too. Consider the meaning of /eat/ again. It has, as we agreed, three slots: that of the subject, the object and the time point. When we want to specify any one of the arguments we must know which one that is. If we want to say who is eating we must be able to connect the subject expression with the appropriate subject slot in the predicate. In predicate logic this mechanism is ensured through a linear notation. That there is eating of a sandwich by Steven at noon today is rendered in relational notation as follows:

$$(4.8) \quad \text{eat}(\text{Steven}, x, 12 : 00) \wedge \text{sandwich}(x)$$

Recall that we agreed to read this existentially: it means that there is a value, say  $s_1$ , for  $x$  that is a sandwich and such that Steven eats it at 12:00. The order of the three arguments, “Steven”, “ $x$ ” and “12:00” is syntactic: the linear alignment in the formula allows to assign them a particular slot in the relation. One may disagree and claim that it is not the notation that achieves this but rather the denotation: /eat/ denotes a three place relation, which in turn is a set of triples. If this is so then we must ask what reality there is to these triples. In predicate logic, it turns out, they have *no reality*. Compare the following pieces of notation:

$$(4.9) \quad p(x, y, z) \quad p(\langle x, y, z \rangle)$$

On the left we have a ternary predicate  $p$ , and three arguments. On the right we have a unary predicate  $p$  being applied to a single argument, the triple  $\langle x, y, z \rangle$ . Standard models for predicate logic do not assume that triples exist. It is true that the interpretation of relation symbols is given in the form of sets of tuples, but these objects are not part of the domain. (First order set theory provides a notable exception to this.) Technically, it is possible to install a domain for such tuples; however, that seems to be a mere technical trick we are pulling. The fundamental question to be asked is namely what makes the arguments come in that particular order as opposed to another. I do not know of any reason to put the subject first. But what is the significance of being the first member in the sequence anyway? I know of no answer to that question. At best, the significance is not objective but rather an artefact of the way we code meanings in predicate logic; this in turn is simply a result of the language we speak. I am sure that speakers of an OSV language would use a different encoding. But what difference would that make in terms of the meaning as opposed to the encoding? In fact, [Dixon, 1994] translates Dyirbal verbs in active morphology by their passive counterparts in English. [Mel’čuk, 1988] goes one step further and says that in Dyirbal the syntactic subject is what is the object of the corresponding English verb.

Now, if it is possible to systematically exchange the first and the second position in the predicate logic encoding then we know that what counts is not the actual position. Rather, what is first in one notation is second in the other, and vice versa. Thus, if the meanings had these positions in them it should not be possible to exchange the positions in this way. This avenue is to be explored. Suppose we have a language just like English except that in transitive constructions all objects and subjects are exchanged. Such a language is not so outlandish: it would be the consistent ergative counterpart of English. Call this language *Erglish*. Thus, for Dixon, Dyirbal is Erglish though with somewhat different pronunciation. The

question is: to what extent can semantics tell the difference between English and English? The answer is: it precisely depends on whether it can tell the difference between being subject and being object. Unless there is a semantic difference, these languages look semantically exactly the same. It therefore appears that if subjects and objects are different we ought to define our semantic rules in terms of this semantic difference rather than an arbitrary label.

Kit Fine has argued in [Fine, 2000] that from a metaphysical point of view we should better renounce the idea of a positionalist view of relations. The calculus of concepts below is an attempt to provide such an account. It will do more than that, as we believe there is more to the problem. Ultimately, we want to say that a property is true not of a sequence (as in predicate logic), nor of a multiset, but rather of a *set* of objects under a particular way of relating the members to a slot. This means that we shall also eliminate *repetitions* in the sequence. It will follow that the concept of self-loving is different from the concept of loving someone else in that the first is unary and the second is binary.

## 4.2 Predicate Logic

Standard semantic theories assume that meanings are adequately described using predicate logic, first or higher order. In this section I shall describe two semantics for many sorted predicate logic. The present section does not introduce predicate logic as an interpreted language; we leave that topic to Section 5.1. In this section we shall concentrate on standard predicate logic, clarify the basic terminology and definitions.

We assume that basic objects are sortal; we have, for example, *objects*, *time points*, *degrees*, *events*, *situations*, *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  $S$  of **sorts**. The sort  $s$  is interpreted by a set  $M_s$ . Thus we have a family of sets  $\mathcal{M} := \{M_s : s \in S\}$ . A **relational type** is a member of  $S^*$ , that is, it is a string of sorts. For a relational type  $\vec{s}$ , a **object of type  $\vec{s}$**  is an element of the set  $M_{\vec{s}}$ , which is defined inductively

as follows.

$$(4.10) \quad \begin{aligned} M_{\langle \rangle} &:= \{\emptyset\} \\ M_{\langle s \rangle} &:= M_s \\ M_{\vec{s}, t} &:= M_{\vec{s}} \times M_t \end{aligned}$$

Finally, a **relation of type**  $\vec{s}$  is a set of objects of type  $\vec{s}$ . The type  $\langle \rangle$  is of 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. The way they do this is somewhat unorthodox. A predicate is true in a model if it has a satisfying tuple (see Definition 4.3). Otherwise it is false. Thus, it is true if its extension is not empty and false otherwise. So, denotations of predicates of type  $\vec{s}$  are subsets of  $M_{\vec{s}}$ . Applied to  $\vec{s} = \langle \rangle$  this gives the desired correspondence.

I also mention that functions are treated basically as relations; a function of type  $\langle s_0, s_1, \dots, s_n \rangle$  is interpreted as follows. Its arguments are of sort  $s_i$ ,  $i < n$ , and the value is of sort  $s_n$ . It is known that we can eliminate functions from a first-order signature (see [Monk, 1976]), and so for simplicity we shall assume that there are no functions.

A **first-order signature** over  $S$  is a pair  $\tau = \langle \text{Rel}, \tau \rangle$  such that Rel is a finite set, the set of **relation symbols** and  $\Xi : \text{Rel} \rightarrow S^*$  an assignment of relational types to relation symbols. Even though  $\tau$  can in principle be infinite, this is excluded here. The alphabet of  $\text{PL}_\tau$  consists of the following symbols

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

$\text{PL}_\tau$  is infinite even if  $\tau$  is finite. This will require creating a new type of index, which is generated from a finite alphabet. From these symbols we can form formulae in the following way:

1. If  $\vec{s} \in \tau(R)$  and  $\vec{x}$  is a sequence of variables of type  $\vec{s}$  then  $R(\vec{x})$  is an **atomic formula**.



2. If  $\varphi$  and  $\chi$  are formulae, so are  $\neg\varphi$ ,  $\varphi \wedge \chi$ ,  $\varphi \vee \chi$  and  $\varphi \rightarrow \chi$ .
3. If  $\varphi$  is a formula and  $x_s^i$  a variable then  $\exists x_s^i.\varphi$  and  $\forall x_s^i.\varphi$  is a formula.

Notice that formulae have no type (or, more accurately, are all of the same type). For each  $s \in S$  there is an identity  $=^s$ , which we normally write  $=$ . Identity is *sortal*;  $x_i^t =^s x_j^u$  is true only if  $t = u = s$  (that is, if the types are identical). 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  $\mathcal{M}$  or a **valuation** 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  $s \in S$ :  $\beta(x_i^s) \in M_s$ . The pair  $\langle \mathcal{M}, \beta \rangle$  is called a  $\tau$ -**model**. 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 a formula  $\varphi$  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.

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

This formulation makes predicate logic amenable to the treatment of this book. Standardly, however, one prefers a different formulation. Let  $\beta$  be a valuation and  $\varphi$  a formula. Then say that  $\varphi$  is **true in  $\mathcal{M}$  under the assignment  $\beta$**  and write  $\langle \mathcal{M}, \beta \rangle \vDash \varphi$ , if  $\beta \in [\varphi]_{\mathcal{M}}$ . This notion is defined inductively by

$$\begin{aligned}
 \langle \mathcal{M}, \beta \rangle \vDash \varphi(\vec{x}) &:\Leftrightarrow \beta(\vec{x}) \in \mathcal{J}(R) \\
 \langle \mathcal{M}, \beta \rangle \vDash \neg\varphi &:\Leftrightarrow \text{not } \langle \mathcal{M}, \beta \rangle \vDash \varphi \\
 \langle \mathcal{M}, \beta \rangle \vDash \varphi \wedge \chi &:\Leftrightarrow \langle \mathcal{M}, \beta \rangle \vDash \varphi \text{ and } \langle \mathcal{M}, \beta \rangle \vDash \chi \\
 (4.12) \quad \langle \mathcal{M}, \beta \rangle \vDash \varphi \vee \chi &:\Leftrightarrow \langle \mathcal{M}, \beta \rangle \vDash \varphi \text{ or } \langle \mathcal{M}, \beta \rangle \vDash \chi \\
 \langle \mathcal{M}, \beta \rangle \vDash \varphi \rightarrow \chi &:\Leftrightarrow \langle \mathcal{M}, \beta \rangle \not\vDash \varphi \text{ or } \langle \mathcal{M}, \beta \rangle \vDash \chi \\
 \langle \mathcal{M}, \beta \rangle \vDash (\exists y)\varphi &:\Leftrightarrow \text{for some } \beta' \sim_y \beta: \langle \mathcal{M}, \beta' \rangle \vDash \varphi \\
 \langle \mathcal{M}, \beta \rangle \vDash (\forall y)\varphi &:\Leftrightarrow \text{for all } \beta' \sim_y \beta: \langle \mathcal{M}, \beta' \rangle \vDash \varphi
 \end{aligned}$$

For a formula  $\varphi$  the set of **free variables**,  $\text{fr}(\varphi)$  is defined as follows.

$$\begin{aligned}
 \text{fr}(R(\vec{y})) &:= \{y_i : i < \text{card}(\tau(R))\} \\
 \text{fr}(\neg\varphi) &:= \text{fr}(\varphi) \\
 \text{fr}(\varphi \wedge \chi) &:= \text{fr}(\varphi) \cup \text{fr}(\chi) \\
 (4.13) \quad \text{fr}(\varphi \vee \chi) &:= \text{fr}(\varphi) \cup \text{fr}(\chi) \\
 \text{fr}(\varphi \rightarrow \chi) &:= \text{fr}(\varphi) \cup \text{fr}(\chi) \\
 \text{fr}((\exists y)\varphi) &:= \text{fr}(\varphi) - \{y\} \\
 \text{fr}((\forall y)\varphi) &:= \text{fr}(\varphi) - \{y\}
 \end{aligned}$$

**Proposition 4.1 (Coincidence Lemma)** *Let  $\beta$  and  $\beta'$  be valuations such that for all  $y \in \text{fr}(\varphi)$   $\beta(y) = \beta'(y)$ . Then  $\langle \mathcal{M}, \beta \rangle \models \varphi$  iff  $\langle \mathcal{M}, \beta' \rangle \models \varphi$ . Alternatively,  $\beta \in [\varphi]_{\mathcal{M}}$  iff  $\beta' \in [\varphi]_{\mathcal{M}}$ .*

A **theory** (or **deductively closed set**) in the signature  $\tau$  is a set of formulae  $T \subseteq L_{\tau}$  such that

for every formula  $\varphi$  and every formula  $\chi$ : if  $\varphi \rightarrow \chi \in T$  and  $\varphi \in T$   
then  $\chi \in T$

There is a calculus for predicate logic, whose nature we shall not elucidate (however, see [Monk, 1976] or [Rautenberg, 2006]). It specifies a relation  $\Delta \vdash \varphi$  between sets  $\Delta$  of formulae and a single formula. With respect to this calculus, we say that  $T$  is **consistent** if for  $\perp := (\exists y)\neg(y = y)$  (any choice of  $y$ ) we do *not* have  $T \vdash \perp$ .

**Theorem 4.2 (Completeness of Predicate Logic)** *For every consistent theory  $T$  there is a model  $\mathcal{M}$  and a valuation  $\beta$  such that for all  $\delta \in T$ :  $\langle \mathcal{M}, \beta \rangle \models \delta$ .*

Instead of interpreting a formula by sets of assignments, an alternative is to use finitary relations. Since this gets us closer to our final interpretation (via concepts), let us see how this approach might go. We assume that we have a slightly different enumeration of the variables as before. Instead of enumerating the variables of each sort separately, we enumerate *all* variables. The set of variables of all sorts

is  $V := \{x_i : i \in \mathbb{N}\}$ . Each of the  $x_i$  has its sort,  $s_i$ , which we leave implicit in the notation. For every formula  $\varphi$  we define the meaning to be a relation  $(\llbracket \varphi \rrbracket)_{\mathcal{M}}$ . Before we specify the precise nature of this relation we shall introduce an idea by Kleene. Let the syntactic objects be pairs  $(\varphi, \vec{x})$ , where  $\varphi$  is a formula and  $\vec{x}$  a sequence of variables. Then we let its denotation be the set of all tuples  $\vec{a}$  of same type as  $\vec{x}$  such that there is a valuation which satisfies  $\varphi$  and sends  $x_i$  to  $a_i$ . For example,  $(x_0 + x_1 = x_3, x_0)$  is a syntactic object and denotes over the natural numbers the set  $\{\langle i \rangle : i \in \mathbb{N}\}$ ;  $(x_0 + x_1 = x_3, x_0, x_3)$  is a syntactic object and it denotes the set  $\{\langle i, j \rangle : i \leq j\}$ . Finally,  $(x_0 + x_1 = x_3, x_0, x_3, x_1)$  denotes the set  $\{\langle i, j, k \rangle : i + k = j\}$ . Call a syntactic object  $(\varphi, \vec{x})$  **complete** if every free variable of  $\varphi$  is in  $\vec{x}$ . (We may or may not disallow repetition of variables.) It is possible to give a compositional semantics for complete syntactic objects (see the exercises).

The problem with predicate logic is that our strings are not pairs of formulae and variables. But there is in fact no need to assume that. Namely, all we need to assume is a canonical linear order on the variables. We then assume that the meaning of the formula  $\varphi$  is what the meaning of  $(\varphi, \vec{x})$  is, where  $\vec{x}$  is a specific set containing the set of free variables of  $\varphi$  in canonical order. The sequence we choose here is  $\langle x_0, x_1, \dots, x_{n-1} \rangle$  where  $x_{n-1}$  is the highest free variable of  $\varphi$ . Thus the relation codes the assignment of the first  $n$  variables  $x_i$ ,  $i < n$ , in the following way. For a valuation  $\beta$  we define the partialisation  $\beta_n := \beta \upharpoonright V_n$ , where  $V_n = \{x_i : i < n\}$  for some  $n$ . We translate the valuation  $\gamma$  into a sequence

$$(4.14) \quad (\beta_n)^\heartsuit := \langle \beta_n(x_i) : i < n \rangle \in \prod_{i < n} M_{s_i}$$

Let  $\ell(\varphi)$  be the largest number such that  $x_{\ell(\varphi)-1} \in \text{fr}(\varphi)$ . Then put

$$(4.15) \quad (\llbracket \varphi \rrbracket)_{\mathcal{M}} := \{(\beta_{\ell(\varphi)})^\heartsuit : \beta \in [\varphi]_{\mathcal{M}}\}$$

Clearly,

$$(4.16) \quad (\llbracket \varphi \rrbracket)_{\mathcal{M}} \subseteq \prod_{i < n} M_{s_i}$$

Now, instead of defining  $(\llbracket \varphi \rrbracket)_{\mathcal{M}}$  via the set of satisfying valuations we can also give an inductive definition. Let  $R^{\rightarrow k}$  be the expansion of  $R$  to a  $k$ -ary relation. This is defined as follows. (a) If  $k$  is less than or equal to the length of  $R$  then  $R^{\rightarrow k} := R$ . (b) If  $k$  is greater than the length of  $R$  then  $R^{\rightarrow k+1} := (R^{\rightarrow k}) \times M_{s_k}$ , where  $s_k$  is the sort of  $x_k$ . For a tuple  $\vec{a}$  let  $[i : b]\vec{a}$  denote the result of replacing  $a_i$  by  $b$ . Given a relation  $R$  of length  $n$ , put

$$(4.17) \quad C_{i.R} := \begin{cases} R & \text{if } i \geq n \\ \{[i : b]\vec{a} : b \in M_{s_i}, \vec{a} \in R\} & \text{else} \end{cases}$$

Finally, let  $V^k$  be the total relation of length  $k$ .

$$\begin{aligned}
 (4.18) \quad \langle R(x_{i_0}, \dots, x_{i_{n-1}}) \rangle_{\mathcal{M}} &:= \{ \vec{a} : \langle a_{i_0}, \dots, a_{i_{n-1}} \rangle \in \mathcal{J}(R) \} \\
 \langle \neg \varphi \rangle_{\mathcal{M}} &:= V^{\ell(\varphi)} - \langle \varphi \rangle_{\mathcal{M}} \\
 \langle \varphi \wedge \chi \rangle_{\mathcal{M}} &:= \langle \varphi \rangle_{\mathcal{M}}^{\rightarrow \ell(\chi)} \cap \langle \chi \rangle_{\mathcal{M}}^{\rightarrow \ell(\varphi)} \\
 \langle \varphi \vee \chi \rangle_{\mathcal{M}} &:= \langle \varphi \rangle_{\mathcal{M}}^{\rightarrow \ell(\chi)} \cup \langle \chi \rangle_{\mathcal{M}}^{\rightarrow \ell(\varphi)} \\
 \langle \varphi \vee \chi \rangle_{\mathcal{M}} &:= (V^{\ell(\chi)} - \langle \varphi \rangle_{\mathcal{M}}^{\rightarrow \ell(\chi)}) \cup \langle \chi \rangle_{\mathcal{M}}^{\rightarrow \ell(\varphi)} \\
 \langle \exists x_i \varphi \rangle_{\mathcal{M}} &:= C_i \cdot \langle \varphi \rangle_{\mathcal{M}}
 \end{aligned}$$

**Example 51.** It is worthwhile to mention a few facts about how we intend to use this for natural language. First, we assume that the denotation of expressions is a relation of some sort. To make this come about, we must eliminate all functions and constants. This technique is known (see [Monk, 1976]). We show some cases. The denotation of /John/ is the set of things being identical to John; we can represent this by the formula  $x = j$ , where  $j$  is the constant denoting John. There is no saturation; merge corresponds to conjunction. The sentence “John left.” contains two pieces whose meaning we can paraphrase as “someone is John” and “someone left”. The syntagma adds the meaning that the two people are the same.



In order to implement the previous idea it is necessary to revise our notion of satisfaction.

**Definition 4.3** We say that  $\mathcal{M} \models \varphi(\vec{x})$  if  $\langle \beta(x_i) : i < n \rangle \in \langle \varphi \rangle_{\mathcal{M}}$  for some  $\beta$ .

For comparison we shall say a few words about the type theoretic interpretation chosen by Montague. Instead of using a “flat” types he introduces a hierarchy as follows. A **functional type** is (a) either an element of  $s$ , or (b) a sequence  $\rightarrow s_0 s_1$  where  $s_0$  and  $s_1$  are functional types. We use variables  $\alpha, \beta$  to denote functional types and also write  $\alpha \rightarrow \beta$  rather than using Polish Notation, to keep within the standard notation. We associate with  $\alpha \rightarrow \beta$  the set of all functions from  $M_\alpha$  to  $M_\beta$ . Montague uses  $e$  for *objects* and  $t$  for *truth values*. A relational type  $\langle s_0, s_1, \dots, s_{n-1} \rangle$  is coded as a functional type

$$(4.19) \quad s_0 \rightarrow (s_1 \rightarrow (\dots \rightarrow (s_{n-1} \rightarrow t)))$$

This allows to dispense with the original ‘flat’ types.

**Exercise 43.** Prove the Coincidence Lemma (Proposition 4.1).

**Exercise 44.** Spell out a compositional approach to the semantics of complete syntactic objects. (You may consult Section 5.1 on this, but the solution should be clear anyhow.)

**Exercise 45.** Show that there is no compositional semantics for syntactic objects in general. (So, dropping the completeness requirement will not work.)

**Exercise 46.** Give an example to show why the semantics  $\langle \varphi \rangle_{\mathcal{M}}$  cannot simply be based on the pairs  $(\varphi, \vec{x})$  where  $\vec{x}$  is exactly the set of free variables of  $\varphi$  in canonical order.

## 4.3 Concepts

Standard semantic theories assume that meanings are adequately described using predicate logic, first or higher order. In this section, however, I shall sketch a different theory of meaning, which is based on *concepts*. 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  $\pi : n \rightarrow n$  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  $\vec{s} \cdot t$  is an **expansion** of  $\vec{s}$ . Given a relation  $R$  of type  $\vec{s}$ , define

$$(4.20) \quad \pi[R] := \{ \pi(\vec{x}) : \vec{x} \in R \}$$

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

$$(4.21) \quad E(R) := \{ \langle x_0, x_1, \dots, x_{n-1}, x_{n-1} \rangle : \langle x_0, x_1, \dots, x_{n-1} \rangle \in R \}$$

This is a relation of type  $\vec{s} \cdot s_{n-1}$ . A relation  $R'$  is said to be a **diagonal expansion** of  $R$  if and only if it has the form  $E(R)$ . Finally, set

$$(4.22) \quad P_t(R) := \{ \langle x_0, x_1, \dots, x_{n-1}, x_n \rangle : \langle x_0, x_1, \dots, x_{n-1} \rangle \in R, x_n \in M_t \}$$

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

**Definition 4.4**  $R'$  is an **immediate variant** of  $R$  if and only if  $R'$  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  $\langle R_i : i < n + 1 \rangle$  such that  $R_0 = R$ ,  $R_n = R'$  and for each  $i < n$ ,  $R_{i+1}$  is an immediate variant of  $R_i$ . We write  $R \sim R'$  if  $R'$  is a variant of  $R$ .

**Example 52.** Let  $S := \langle \ell, n \rangle$ ,  $M_\ell := \{a, b, c\}$  and  $M_n := \{0, 1\}$ . The relation  $R = \{\langle a, 0 \rangle, \langle b, 1 \rangle\}$  is of type  $\langle \ell, n \rangle$ . 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

$$(4.23) \quad E_i(R) := \{\langle x_0, x_1, \dots, x_{n-1}, x_i \rangle : \langle x_0, x_1, \dots, x_{n-1} \rangle \in R\}$$

Then  $E_i(R)$  is a variant of  $R$ . Namely, let  $\pi = (i \ n - 1)$  (see Appendix for notation) 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

$$(4.24) \quad E_i(R) = \pi'[E(\pi[R])]$$

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

$$(4.25) \quad P_t^i(R) := \{\langle x_0, x_1, \dots, x_{n-1}, x_n \rangle : \langle x_0, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_{n-1} \rangle \in R, x_i \in M_t\}$$



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

$$(4.26) \quad \{\langle x, x \rangle : x \in M_s\}$$

This is a diagonal 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  $M_\emptyset = \{\emptyset\} = 1$ . Thus the identity relation is a variant of the “true” relation. This has consequences we shall look at in more detail later.

**Definition 4.5** A *concept* is a set of relations of the form  $\llbracket R \rrbracket := \{R' : R' \sim R\}$ . Concepts are denoted by small Fraktur letters:  $cc$ ,  $cd$ .

In principle we should write  $\llbracket R \rrbracket_{\mathcal{U}}$  since the concept depends on the structure; however, I shall drop the reference to the structure since it will always be clear from the context. There are two special concepts: the *verum* concept, denoted by  $t$ , and the *falsum* concept, denoted by  $f$ . We have

$$(4.27) \quad t := \llbracket M \rrbracket, \quad f := \llbracket \emptyset \rrbracket$$

We employ the following convention. For a set  $M$  we take  $M$  to be the same as  $1 \times M$ , where  $1 = \{\emptyset\}$ . Thus, if  $M_s$  is the domain of elements of type  $s$ , since  $M_s$  and  $1 \times M_s$  count as the same, the set (= relation)  $M_s$  is a variant of 1. This is to be kept in mind.  $M^1 = \{\langle x \rangle : x \in M\}$  is technically different from  $M$ , but considered here the same object.

**Example 53.** The concept generated by the empty relation is of course just the set  $t := \{\emptyset\}$ . This is the falsum concept. The verum concept is the concept of the form  $t = \llbracket \{\emptyset\} \rrbracket$ . If the universe has just one member, say  $\{a\}$ , then these are the only two concepts. For let  $R$  be a nonempty relation. Then it has the form  $\{\langle a, a, \dots, a \rangle\}$ . Any two such sets are variants of each other. For example,  $\{\langle a, a, a \rangle\}$  is a variant of  $\{\langle a, a \rangle\}$ , which in turn is a variant of  $\{\langle a \rangle\}$ . The latter is a variant of 1. Thus, every nonempty relation is a variant of every other nonempty relation, but not a variant of the empty relation.  $\otimes$

**Example 54.** We shall describe the concepts over a two element universe  $M := \{a, b\}$ . The zeroary relations are  $\emptyset$  and  $\{\emptyset\}$ , generating the concepts  $t$  and  $f$ . The unary relations are  $\emptyset$ ,  $\{\langle a \rangle\}$ ,  $\{\langle b \rangle\}$ ,  $M = \{\langle a \rangle, \langle b \rangle\}$ . The first and the last are variants of zeroary relations, so we effectively have only two new members,  $\{\langle a \rangle\}$  and  $\{\langle b \rangle\}$ .

Next we turn to binary relations. Here is a list of all 16:

$$(4.28) \quad \begin{array}{ll} R_1 := \emptyset & R_9 := \{\langle a, b \rangle, \langle b, a \rangle\} \\ R_2 := \{\langle a, a \rangle\} & R_{10} := \{\langle a, b \rangle, \langle b, b \rangle\} \\ R_3 := \{\langle a, b \rangle\} & R_{11} := \{\langle b, a \rangle, \langle b, b \rangle\} \\ R_4 := \{\langle b, a \rangle\} & R_{12} := \{\langle a, a \rangle, \langle a, b \rangle, \langle b, a \rangle\} \\ R_5 := \{\langle b, b \rangle\} & R_{13} := \{\langle a, a \rangle, \langle a, b \rangle, \langle b, b \rangle\} \\ R_6 := \{\langle a, a \rangle, \langle a, b \rangle\} & R_{14} := \{\langle a, a \rangle, \langle b, a \rangle, \langle b, b \rangle\} \\ R_7 := \{\langle a, a \rangle, \langle b, a \rangle\} & R_{15} := \{\langle a, b \rangle, \langle b, a \rangle, \langle b, b \rangle\} \\ R_8 := \{\langle a, a \rangle, \langle b, b \rangle\} & R_{16} := \{\langle a, a \rangle, \langle a, b \rangle, \langle b, a \rangle, \langle b, b \rangle\} \end{array}$$

$R_1$  and  $R_{16}$  are variants of  $\emptyset$  and  $\{\emptyset\}$ , respectively.  $R_2$  and  $R_5$  are diagonal expansions of  $\{\langle a \rangle\}$  and  $\{\langle b \rangle\}$ , respectively.  $R_3$  and  $R_4$  are permutations of each other.  $R_6$  is  $\{\langle a \rangle\} \times M$ , so it is a variant of  $\{\emptyset\}$ ;  $R_7$  is a permutation of  $R_6$ .  $R_8$  is the identity on  $M$ , hence in turn a variant of verum.  $R_9$  is symmetrical; it generates a concept different from the previous.  $R_{10}$  and  $R_{11}$  are diagonal expansions of unary relations.  $R_{12}$ ,  $R_{13}$  and  $R_{15}$  are essentially new, while  $R_{14}$  is a variant of  $R_{13}$ . Thus, up to variants, there are only six relations:  $R_3$ ,  $R_6$ ,  $R_9$ ,  $R_{12}$ ,  $R_{13}$  and  $R_{15}$ .  $\odot$

Notice that the empty set is the empty  $n$ -ary relation for every  $n$ . It thus plays multiple roles. This is not so for concepts. The empty concept has length 0. The empty binary relation generates the empty concept, just as any other empty relation, since they are the same set.

It is to be noted that identity, which plays such a big role in predicate logic, denotes the diagonal  $\Delta_M := \{\langle a, a \rangle : a \in M\}$ . This set is the diagonal expansion of  $M$ . Hence identity is a variant of 1, and therefore generates the concept  $t$ . This reflects the fact that self-identity is trivially true of everything. To say that an object is identical to itself is to issue a mere triviality. For this it does not matter whether or not we take identity to be sortal. For example, the sortal diagonal  $\Delta_s := \{\langle a, a \rangle : a \in M_s\}$  is a diagonal expansion of  $M_s$ , which is an expansion of 1.

Let us now investigate the structure of the concept space somewhat. From now on concepts are denoted by Gothic letters, such as  $\mathfrak{c}$ ,  $\mathfrak{d}$  and so on.

**Definition 4.6** *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  $\mathfrak{c}$ . The **length** of  $\mathfrak{c}$  is the length of any minimal member of  $\mathfrak{c}$ . The length of  $\mathfrak{c}$  is denoted by  $\ell(\mathfrak{c})$ .*



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

**Proposition 4.7** *Let  $R$  and  $R'$  be minimal members of a concept  $c$ . 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  $h : n \rightarrow k$  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  $R$ .

The map  $h$  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

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

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.

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

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.

**Lemma 4.8** *Let  $R, R'$  be minimal members of  $c$ . If  $R \subseteq R'$  then  $R = R'$ .*

**Proof.** Suppose that  $R \subseteq R'$ . By the previous theorem,  $R' = \pi[R]$  for some permutation  $\pi$ . So,  $R \subseteq \pi[R]$ . From this we derive  $\pi^i[R] \subseteq \pi^{i+1}[R]$  for any  $i$ , and by transitivity,  $R \subseteq \pi^i[R]$  for any  $i$ . Now, since there is a  $k$  such that  $\pi^k$  is the identity, we can also derive  $\pi^{k-1}[R] \subseteq \pi^k[R] = R$ , and reasoning backwards establish that  $\pi^i[R] \subseteq R$  for all  $i < k$ . It follows that  $R' = \pi[R] \subseteq R$ .  $\square$

We can use this to define the type of a concept. Suppose  $cc$  is a concept and that  $R \in cc$  is minimal. Then  $R$  has a type  $\vec{s}$ . This is a sequence. It defines a multiset  $\S(\vec{s})$  in the following way: the sort  $s$  is contained in  $\S(\vec{s})$  exactly as many times as it is contained in  $\vec{s}$ . Thus we say that  $\S(\vec{s})$  is the **type** of  $cc$ .

We define the following subsumption relation on concepts.

$$(4.31) \quad c \leq d : \Leftrightarrow (\forall R \in c)(\exists S \in d)(R \subseteq S)$$

Notice that  $R \subseteq S$  means that the relations are of same length and type. It turns out that just one pair of sets is sufficient to establish an order between the concepts.

**Lemma 4.9**  *$c \leq d$  if and only if there is  $R \in c$  and  $S \in d$  such that  $R \subseteq S$ .*

**Proof.** Let  $R \in c$  and  $S \in d$  be such that  $R \subseteq S$ . Let  $\pi$  be a permutation. Then  $\pi[R] \subseteq \pi[S]$ . Also,  $R \times M \subseteq S \times M$  and  $E(R) = E(S)$ . So for any permutation

and expansion of  $R$  there is a corresponding set in  $\mathfrak{d}$ . If however  $R$  is itself an expansion of  $T$  then  $T = C_i.R$  for some  $i$ . Now,  $C_i.R \subseteq C_i.S$ . Hence for all  $R' \sim R$  there is a  $S' \sim S$  such that  $R' \subseteq S'$ .  $\square$

**Proposition 4.10**  $\leq$  is an ordering relation. That is to say for all  $\mathfrak{c}, \mathfrak{d}$  and  $\mathfrak{e}$ :

- ①  $\mathfrak{c} \leq \mathfrak{c}$ ,
- ② if  $\mathfrak{c} \leq \mathfrak{d}$  and  $\mathfrak{d} \leq \mathfrak{e}$  then  $\mathfrak{c} \leq \mathfrak{e}$ , and
- ③ if  $\mathfrak{c} \leq \mathfrak{d}$  and  $\mathfrak{d} \leq \mathfrak{c}$  then  $\mathfrak{c} = \mathfrak{d}$ .

**Proof.** ① is clear. For ②, suppose  $R \in \mathfrak{c}$ . Then by assumption there is  $S \in \mathfrak{d}$  such that  $R \subseteq S$ ; again by assumption there is a  $T \in \mathfrak{e}$  such that  $S \subseteq T$ . So,  $R \subseteq T$  for some  $T \in \mathfrak{e}$ . For ③ let  $R$  be minimal in  $\mathfrak{c}$ . Assume first that there is a minimal  $S \in \mathfrak{d}$  such that  $R \subseteq S$ . Then by assumption there is a  $R' \in \mathfrak{c}$  such that  $S \subseteq R'$ . Since  $R \subseteq R'$ , and both are of same length,  $R'$  is not only minimal (by Proposition 4.7), we also have  $R = R'$ , by Lemma 4.8. It follows that  $R = S$ , and  $\mathfrak{c} = \mathfrak{d}$ . Now suppose that there is no minimal  $S$  such that  $R \subseteq S$ . Then  $\mathfrak{d}$  has lesser length than  $\mathfrak{c}$ , for there is at least one  $S$  of length  $\ell(\mathfrak{c})$  in  $\mathfrak{d}$ . Hence  $\ell(\mathfrak{d}) < \ell(\mathfrak{c})$ . Now pick a minimal  $S \subseteq \mathfrak{d}$ . There is no  $R \in \mathfrak{c}$  for which  $S \subseteq R$ , contrary to assumption.  $\square$

The concatenation of concepts plays the role of conjunction.

**Definition 4.11** Suppose that  $\mathfrak{c} = \llbracket R \rrbracket$  and  $\mathfrak{d} = \llbracket S \rrbracket$ . Then we define

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

This definition does not depend on representatives. We omit the proof. Notice that even if  $R$  is minimal in  $\mathfrak{c}$  and  $S$  is minimal in  $\mathfrak{d}$ ,  $R \times S$  need not be minimal in  $\mathfrak{c} * \mathfrak{d}$ . This is easily seen if  $\mathfrak{c} = \mathfrak{d}$ .

**Proposition 4.12**  $*$  is a semilattice operation. This means that for all  $\mathfrak{c}, \mathfrak{d}$  and  $\mathfrak{e}$ :

- ①  $\mathfrak{c} * \mathfrak{c} = \mathfrak{c}$ .
- ②  $\mathfrak{c} * \mathfrak{d} = \mathfrak{d} * \mathfrak{c}$ .

$$\textcircled{3} \quad c * (d * e) = (c * d) * e.$$

**Proof.** Let  $c = \llbracket R \rrbracket$ ,  $d = \llbracket S \rrbracket$  and  $e = \llbracket T \rrbracket$ . Then, as  $R \times R \sim R$  (using a series of diagonal expansions), we have  $\llbracket R \times R \rrbracket = \llbracket R \rrbracket = c$ . Further, since  $R \times S \sim S \times R$  (using a suitable permutation) we have  $c * d = d * c$ . Finally,  $(c * d) * e$ , since  $R \times S \in c * d$ , and so  $(c * d) * e = \llbracket (R \times S) \times T \rrbracket = \llbracket R \times (S \times T) \rrbracket = c * (d * e)$ .  $\square$

The concatenation is a kind of conjunction. It represents the conjunction without any identification. In fact we can show that under the ordering  $\leq$  defined above,  $*$  is exactly the greatest lower bound.

**Proposition 4.13**  $c * d \leq c$  and  $c * d \leq d$ . Moreover, for every  $e$  such that  $e \leq c$  and  $e \leq d$  we also have  $e \leq c * d$ .

**Proof.** The first assumption follows from the second. Assume therefore  $e \leq c$  and  $e \leq d$  for some  $e$ . Pick  $R \in e$ . There is then  $S \in c$  and  $T \in d$  such that  $R \subseteq S$  and  $R \subseteq T$ . Let  $R$  be of length  $n$ . Define the set  $R^{\text{pd}}$  as follows.

$$(4.33) \quad R^{\text{pd}} := \{\langle a_0, \dots, a_{n-1}, a_0, \dots, a_{n-1} \rangle : \langle a_0, \dots, a_{n-1} \rangle \in R\}$$

$R^{\text{pd}} \sim R$  (by repeated generalised diagonal expansion). Moreover,  $R^{\text{pd}} \subseteq S \times T$ . By Lemma 4.9,  $e \leq c * d$ .  $\square$

There is no natural definition of disjunction, since this needs identification of columns. We leave it to the next section to go deeper into the topic of identification of columns across concepts.

Now we shall interpret formulae not by sets of assignments or finitary relations, but by concepts. The definition is as follows.

$$(4.34) \quad \langle\langle\varphi\rangle\rangle_{\mathcal{M}} := \llbracket (\langle\varphi\rangle)_{\mathcal{M}} \rrbracket$$

Recall that  $(\langle\varphi\rangle)_{\mathcal{M}}$  delivers a relation (a subset of  $\prod_{i < \ell(\varphi)} M_{s_i}$ ) based on the set of free variables of  $\varphi$ .

We can give a somewhat more compact version of this set. Notice namely that  $(\langle\varphi\rangle)_{\mathcal{M}}$  was based on a set that may properly include the set  $\text{fr}(\varphi)$ . For if  $x_i$  is not free but there is  $j > i$  such that  $x_j$  is free in  $\varphi$ , then  $\varphi$  does not depend on  $x_i$  but nevertheless the  $i$ th component of  $(\langle\varphi\rangle)_{\mathcal{M}}$  records the values of  $x_i$ . It is thus easily seen that there are sets  $A \subseteq \prod_{j < i} M_{s_j}$  and  $B \subseteq \prod_{i < j < \ell(\varphi)} M_{s_j}$  such that

$$(4.35) \quad (\langle\varphi\rangle)_{\mathcal{M}} \subseteq A \times M_{s_i} \times B$$

There is a set  $C \subseteq A \times B$  such that

$$(4.36) \quad \langle\langle\varphi\rangle\rangle_{\mathcal{M}} = \{\vec{x} \cdot y \cdot \vec{z} : \vec{x} \cdot \vec{z} \in C, y \in M_{s_i}\}$$

By the laws of concepts,

$$(4.37) \quad \llbracket \langle\langle\varphi\rangle\rangle_{\mathcal{M}} \rrbracket = \llbracket C \times M_{s_i} \rrbracket = \llbracket C \rrbracket$$

Thus, we can actually eliminate from  $\langle\langle\varphi\rangle\rangle_{\mathcal{M}}$  all columns referring to variables that are not free in  $\varphi$ .

However, one should not be misled to think that it is exactly the free variables whose values need to be recorded for the formation of the concept. For sometimes variables occur free but nevertheless make no significant contribution to the formula. For example, for the formula  $\chi := \varphi(\vec{y}) \wedge x_k^s = x_k^s$  we get  $\text{fr}(\chi) = \text{fr}(\varphi) \cup \{x_k^s\}$ . If  $k \geq \ell(\varphi)$  we have

$$(4.38) \quad \langle\langle\varphi\rangle\rangle_{\mathcal{M}} \neq \langle\langle\chi\rangle\rangle_{\mathcal{M}}$$

On the other hand we have

$$(4.39) \quad [\varphi]_{\mathcal{M}} = [\chi]_{\mathcal{M}}$$

since both formulae are satisfied by the same assignments. We have  $\langle\langle\chi\rangle\rangle_{\mathcal{M}} = \langle\langle\varphi\rangle\rangle_{\mathcal{M}}$ . Thus the addition of ‘trivial’ variables has no effect on the concept.

Let us finally turn to elementarily definable concepts. Suppose that  $R = \langle\langle\varphi(x_0, \dots, x_{n-1})\rangle\rangle_{\mathcal{M}}$  for some  $\varphi(x_0, \dots, x_{n-1})$ . In this case  $R$  is said to be **definable**. Then

- ①  $\pi[R] = \langle\langle\varphi(x_{\pi(0)}, \dots, x_{\pi(n-1)})\rangle\rangle_{\mathcal{M}}$ .
- ②  $R \times M = \langle\langle\varphi(x_0, \dots, x_{n-1}) \wedge x_n = x_n\rangle\rangle_{\mathcal{M}}$ .
- ③  $E(R) = \langle\langle\varphi(x_0, \dots, x_{n-1}) \wedge x_{n-1} = x_n\rangle\rangle_{\mathcal{M}}$ .

Hence, if one minimal member of a concept is definable, all members of the concept are definable. However we can prove more.

**Proposition 4.14** *Let  $c$  be a concept and  $R, S \in c$ . Then  $R$  is definable if and only if  $S$  is.*

**Proof.** It remains to be shown that if  $E(R)$  or  $R \times M$  is definable, so is  $R$ . To this end, let  $\langle \varphi(x_0, \dots, x_n) \rangle_{\mathcal{M}} = E(R)$ . Then  $\langle \exists x_n. \varphi(x_0, \dots, x_n) \rangle_{\mathcal{M}} = R$ . Similarly, if  $\langle \varphi(x_0, \dots, x_n) \rangle_{\mathcal{M}} = R \times M$  then  $\langle \exists x_n. \varphi(x_0, \dots, x_n) \rangle_{\mathcal{M}} = R$ .  $\square$

Thus the variants of a relation can be obtained through adding some equation or existentially quantifying a relation. 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  $\langle \chi \rangle_{\mathcal{M}} = \langle \varphi \rangle_{\mathcal{M}}$ . It is therefore the case that

$$(4.40) \quad \langle x_0^e < x_1^e \rangle_{\mathcal{M}} = \langle x_0^e > x_1^e \rangle_{\mathcal{M}}$$

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.

**Exercise 47.** Show that  $c \leq d$  does not hold if  $\ell(c) < \ell(d)$ . However, give examples where  $\ell(d) > \ell(c)$  and still  $c \leq d$ .

**Exercise 48.** Show that  $\langle \varphi(x_0, x_1) \wedge x_0 = x_1 \rangle \leq \langle \varphi(x_0, x_1) \rangle$  need not hold.

**Exercise 49.** Show that if  $R \subseteq S$  then  $C_i.R \subseteq C_i.S$  and  $E(R) \subseteq E(S)$ .

## 4.4 Linking Aspects and Constructional Meanings

The previous section has introduced the concatenation of concepts, which turned out to be conjunction in the sense of the ordering. However, when we spell this out in terms of defining formulae we get something slightly different.

**Proposition 4.15** *Let  $\varphi$  and  $\chi$  be formulae. Let  $s$  be an injective substitution such that  $\text{fr}(\varphi) \cap \text{fr}(s(\chi)) = \emptyset$ . Then*

$$(4.41) \quad \langle \varphi \rangle * \langle \chi \rangle = \langle \varphi \wedge s(\chi) \rangle$$

The proof is easy and left as an exercise. We just point out an example to show why it is generally not the case that  $\langle\langle\varphi\rangle * \langle\psi\rangle = \langle\varphi \wedge \psi\rangle$ . Let  $\varphi = x_0 < x_1$  and  $\psi = x_1 < x_0$ . Then  $\varphi \wedge \psi$  is unsatisfiable, hence  $\langle\varphi \wedge \psi\rangle$  is the null concept. On the other hand, the concatenation is not empty, so cannot be the null concept. According to the theorem above it is  $\langle x_0 < x_1 \wedge x_2 < x_3 \rangle$ .

This is a welcome result. [Vermeulen, 1995] has made the point that the merge operation to be employed for merging DRSs should not be done in the style of [Zeevat, 1989], that is, simply taking all variables at face value. Recall that the Zeevat-merge was defined like this, where  $\langle V, \Gamma \rangle$  and  $\langle W, \Delta \rangle$  are pairs of variable sets and sets of formulae:

$$(4.42) \quad \langle V, \Gamma \rangle \bullet \langle W, \Delta \rangle := \langle V \cup W, \Gamma \cup \Delta \rangle$$

One of the problems that this faces is accidental capture:

$$(4.43) \quad \langle \{x\}, \emptyset \rangle \bullet \langle \emptyset, \{\varphi(x)\} \rangle = \langle \{x\}, \{\varphi(x)\} \rangle$$

The left hand sides read “ $\exists x$ ” and “ $\varphi(x)$ ”, respectively, and the right hand side  $\exists x.\varphi(x)$ . Such results can only be obtained by intelligent variable handling. On occasion, though, we really *do* want variables to be identified. This is the case with the phrase /a dog/, which is the concatenation of /a/ and /dog/, which translate as  $\langle \{x\}, \emptyset \rangle$  and  $\langle \emptyset, \{\text{dog}(x)\} \rangle$ , respectively. The result we want is  $\langle \{x\}, \{\text{dog}(x)\} \rangle$ . To get this effect, [Vermeulen, 1995] introduces *names*. Variables are optionally paired with a name, which can be anything, even an index, and those variables that have the same name will be identical after merge.<sup>3</sup> Let  $[x \mapsto 1]$  be the function mapping the variable  $x$  to 1. Then with these stipulations we get

$$(4.44) \quad \begin{aligned} \langle [x \mapsto 1], \langle \{x\}, \emptyset \rangle \rangle \bullet \langle [x \mapsto 1], \langle \emptyset, \{\text{dog}(x)\} \rangle \rangle \\ = \langle [x \mapsto 1], \langle \{x\}, \{\text{dog}(x)\} \rangle \rangle \end{aligned}$$

$$(4.45) \quad \begin{aligned} \langle [x \mapsto 1], \langle \{x\}, \emptyset \rangle \rangle \bullet \langle [x \mapsto 2], \langle \emptyset, \{\text{dog}(x)\} \rangle \rangle \\ = \langle [x \mapsto 1; y \mapsto 2], \langle \{x\}, \{\text{dog}(y)\} \rangle \rangle \end{aligned}$$

In this system the names of the variables are insignificant. Variables can be re-named inside a representation as long as distinct variables are mapped to distinct variables. Yet, the names of the variables are significant in the same way as the

<sup>3</sup>The actual referent systems operated with a pair of such injections, but we can safely ignore that complication.

variable was in the Zeevat-merge. Thus we have made not much progress, because the names cannot be part of the meaning.

What we need to find is a definition of merge that does not assume that the functions are part of the representation. Instead, we must be able to define them on the basis of the concept itself. We show how to transform Vermeulen's approach. First, we simplify it by using numbers in place of names. It is clear that the names can be absolutely anything, since the only thing that matters for merge is whether names are equal or different. Now think of each number as naming a position in a tuple. Then instead of using names to associate with variable, we associate positions in a tuple, and the positions are simply numbers. Same number means then that the variable will be associated with the same position in a tuple. This leads directly to the idea of simply associating a relation with a concept. So the idea is basically this. Assume that  $f$  and  $g$  are functions from concepts to relations such that  $f(c) \in c$  for every  $c$ . Then put

$$(4.46) \quad c \otimes^{f,g} d := \llbracket f(c) \cap g(d) \rrbracket$$

This is well-defined just in case  $f(c)$  and  $g(d)$  are relations of the same type.

**Example 55.** Transitive verbs can be coordinated to form transitive verbs. The meaning of /fry and eat/ is again a 2-concept as witnessed by /fry and eat a sausage/. Let  $f = g$  both be such that they assign to the 2-concept  $\langle \text{fry}'(x_0, x_1) \rangle_{\mathcal{M}}$  the set  $\langle \text{fry}'(x_0, x_1) \rangle_{\mathcal{M}}$ , and similarly to  $\langle \text{eat}'(x_0, x_1) \rangle_{\mathcal{M}}$  the set  $\langle \text{eat}'(x_0, x_1) \rangle_{\mathcal{M}}$ . Then on the basis of this choice,

$$(4.47) \quad \begin{aligned} & \langle \text{fry}'(x_0, x_1) \rangle_{\mathcal{M}} \otimes^{f,g} \langle \text{eat}'(x_0, x_1) \rangle_{\mathcal{M}} \\ &= \llbracket \langle \text{fry}'(x_0, x_1) \rangle_{\mathcal{M}} \cap \langle \text{eat}'(x_0, x_1) \rangle_{\mathcal{M}} \rrbracket_{\mathcal{M}} \\ &= \llbracket \langle \text{fry}'(x_0, x_1) \wedge \text{eat}'(x_0, x_1) \rangle_{\mathcal{M}} \rrbracket_{\mathcal{M}} \\ &= \langle \text{fry}'(x_0, x_1) \wedge \text{eat}'(x_0, x_1) \rangle_{\mathcal{M}} \end{aligned}$$

It is however also possible to coordinate concepts of different length, for example /hit and run/. Here, /hit/ denotes a 2-concept and /run/ a 1-concept. In this connection, /hit/ functions in the same way as /hit someone/. To make this work, we need to select for  $\langle \text{run}'(x_0) \rangle$  not the set  $\langle \text{run}'(x_0) \rangle$  but the set  $\langle \text{run}'(x_0) \rangle \times M$ . Intersect this with the set  $\langle \text{hit}'(x_0, x_1) \rangle$  and one gets the set  $\langle \text{hit}'(x_0, x_1) \rangle \cap \langle \text{run}'(x_0) \rangle$  of pairs  $\langle x, y \rangle$  such that  $x$  hits  $y$  and runs. This is as desired.  $\odot$



As concepts are defined (uniquely) by their minimal member, a slightly different approach to defining  $\mathbb{O}^{f,g}$  is by using a minimal member as a representative of each concept. A linking aspect is a function that does the job of finding such representatives.

**Definition 4.16** A *linking aspect* is a partial function  $Y$  defined on some set of concepts such that  $Y(c)$  is a member of  $c$ .  $Y$  is *minimal* if  $Y(c)$  is a minimal member of  $c$  for every  $c$ .

A particular way to define a linking aspect is by means of critical sets.

**Definition 4.17** Let  $c$  be a concept,  $R$  a minimal member of  $c$ . A *critical set for  $R$*  is a set  $A$  such that for all minimal  $Q \in c$ : if  $A \subseteq Q$  then  $Q = R$ .

Instead of mapping concepts to relations we can map them to critical sets. Let  $V$  be such a map. Then given  $c$ ,  $Y_V(c)$  is defined to be the unique minimal member of  $c$  containing  $V(c)$ .

**Example 56.** Take the concept defined by  $<$  on the natural numbers. It has two minimal members:  $\{\langle i, j \rangle : i < j\}$  and  $\{\langle i, j \rangle : i > j\}$ . The pair  $\langle 0, 1 \rangle$  is in the first and not the second. Therefore  $\{\langle 0, 1 \rangle\}$  is a critical set. Similarly, suppose that John is taller than Phil. Then the concept denoted by “is taller than” has two minimal relations, only one of which contains  $\langle \text{John}, \text{Phil} \rangle$ . Therefore,  $\{\langle \text{John}, \text{Phil} \rangle\}$  is a critical set. ⊛


For a relation  $S$  let  $\Pi(S)$  be the following partition of  $n$ :  $C \in \Pi(S)$  iff for all  $\vec{x} \in S$  and all  $i, j \in C$ ,  $x_i = x_j$ . It is not hard to see that  $A$  is critical for  $R$  iff  $\Pi(A) = \Pi(R)$ . Now,  $\Pi(\emptyset) = \{n\}$ . We now define a  $\vec{x}_i \in R$  as follows. Put  $A_i := \{\vec{x}_j : j < i\}$ . If  $\Pi(A_i) \neq \Pi(R)$  then let  $\vec{x}_i \in R$  be chosen such that one of the sets from  $\Pi(\{\vec{x}_i\})$  is not a join of partition sets from  $\Pi(A_i)$ . Such an element must exist if  $\Pi(A_i) \neq \Pi(R)$ . In that case,  $\Pi(A_{i+1}) \neq \Pi(A_i)$ . Since the size of the partition sets must decrease with every step it is easy to see that we can take only  $n - 1$  steps; that is, we need to choose at most  $n - 1$   $\vec{x}_i$ .

**Proposition 4.18** Let  $c$  be of length  $n$  then for every minimal  $R \in c$  there is a critical set of cardinality at most  $n - 1$ .

This dramatically improves the bound given by [Dorr, 2004] of  $n! - 1$ . This is the best possible result. (We leave a proof of this claim to the exercises.)


**Example 57.** To see that it is not at all a weird idea to consider conjunction to be ambiguous let us look at the notion of a syntactic pivot. In English the following sentence implies that John fell:

(4.48) John kissed the woman and fell.

We say that /John/ is the **pivot** in the coordination. This is ordinarily attributed to the fact that we have a VP coordination, and /John/ is the subject of both. There are languages in which the same coordination will imply that the woman fell. Such languages are invariably ergative (see [Dixon, 1994]); however, it is not the case that ergative languages all function in this way. Thus we need to distinguish between ergativity in case marking and ergativity in pivot choice. Similarly, some languages indicate whether or not a clause uses the same subject. Thus it explicitly marks part of the linking aspect to be used. 

**Example 58.** The linking aspect is responsible for dealing with pronouns.

(4.49) John saw the thief in his office.

The pronoun /his/ may denote either John or the thief or a third person. In the present case we can paraphrase its meaning by “belong to someone”. Thus, the phrase /in his office/ has the meaning “in the office belonging to someone”. We can interpret this someone as John, the thief or leave it unidentified. Again, for that we need different linking aspects if we insist that the only operation we want to use is conjunction. 

Linking aspects give great flexibility in handling coordination. Every concept can be treated independently from the other. This might not be so desirable and leads to results that may be surprising.

**Example 59.** It is possible to define reflexivization of 2-concepts through concept conjunction. Namely, put  $Y(1) = \{\langle x, x \rangle\}$ . Then let  $c$  be a 2-concept with minimal member  $R$ .

(4.50)  $c \otimes^Y 1 = R \cap \{\langle x, x \rangle : x \in M\}$



**Example 60.** Let  $M = \{a, b, c, d\}$ . There are  $c$  and  $Y$  such that  $c \circledast^Y 1 \neq (c \circledast^Y 1) \circledast^Y 1$ . Namely, let  $R = \{\langle a, a, a \rangle, \langle a, a, b \rangle, \langle a, b, a \rangle, \langle a, b, b \rangle, \langle a, a, c \rangle\}$ ,  $c = \llbracket R \rrbracket$ . Further, let  $Y(1) = \{\langle x, x \rangle : x \in M\} \times M$  and  $Y(c) = R$ . Then

$$(4.51) \quad c \circledast^Y 1 = \llbracket \{\langle a, a, a \rangle, \langle a, a, b \rangle, \langle a, a, c \rangle\} \rrbracket = \llbracket \{\langle a, a \rangle, \langle a, b \rangle, \langle a, c \rangle\} \rrbracket$$

Finally, put  $Y(\{\langle a, a \rangle, \langle a, b \rangle, \langle a, c \rangle\}) := \{\langle a, a \rangle, \langle a, b \rangle, \langle a, c \rangle\}$  and we get

$$(4.52) \quad \begin{aligned} (c \circledast^Y 1) \circledast^Y 1 &= (\llbracket \{\langle a, a \rangle, \langle a, b \rangle, \langle a, c \rangle\} \rrbracket) \circledast^Y 1 \\ &= \llbracket \{\langle a \rangle\} \rrbracket \end{aligned}$$



$\circledast^{YZ}$  is unfortunately somewhat inflexible. When we merge  $c$  and  $d$  via  $Y$  and  $Z$ , this is defined only if  $Y(c)$  and  $Z(d)$  have same length. Thus if  $Y(e)$  has different length as  $Y(d)$ , then only one of  $c \circledast^{YZ} d$  and  $c \circledast^{YZ} e$  is defined. Thus this operation is too unflexible. A better version is as follows. Let  $U$  be a function from pairs of concepts to pairs of relations such that if  $U(c, d) = (R, S)$  then  $R \in c$  and  $S \in d$ . Then put

$$(4.53) \quad c \circledast^U d := [R \cap S], \quad \text{where } U(c, d) = (R, S)$$

This function offers more flexibility than might be needed in natural languages, but that is another matter. We conclude with a useful characterization of the logical strength of these operations.

**Proposition 4.19** *Let  $c = \langle\langle \varphi(\vec{x}) \rangle\rangle$  and  $d = \langle\langle \psi(\vec{y}) \rangle\rangle$  with  $\vec{x}$  and  $\vec{y}$  disjoint. Then there is a formula  $\chi$ , which is a conjunction of equations of the form  $x_i = y_j$  such that  $(c \circledast^U d) = \langle\langle \varphi(\vec{x}) \wedge \psi(\vec{y}) \wedge \chi \rangle\rangle$ .*

I conclude this section with a characterisation of the admissible meanings. By an admissible meaning I mean such a meaning that is not provided through a lexical entry but is rather defined by the grammar. In Montague Grammar there was no need to talk about admissible meanings. If a constituent is formed, its meaning is completely determined by the meaning of its two parts. The introduction of concepts, however, has not only made it possible to use different linking aspects (and so to get different resulting meanings). The introduction of linking aspects was actually also necessitated since linking of arguments places is not unique. Additionally, the introduction of new intermediate variables has the drawback of

introducing discourse objects where sometimes none should exist. Thus, we also need a mechanism to remove them. Section 4.7 will introduce a way to do this without removing them. Here we shall revert to the standard way, namely quantification. Thus we generalise the operation (4.53) once more. Let  $H$  be a set of numbers. Define for a relation  $R$  the operation  $C_H.R$  as follows.

$$(4.54) \quad \begin{aligned} C_{\emptyset}.R &:= R \\ C_H.R &:= C_{H-(i)}.C_i.R \end{aligned}$$

In the equations above we assume that  $i$  is actually in  $H$ . (This is not strictly required, but makes the definition well-founded.) The general scheme of constructional meaning is now this.

$$(4.55) \quad c \otimes^{U,H} d := [C_H.(R \cap S)], \quad \text{where } U(c, d) = (R, S)$$

**Exercise 50.** Prove Proposition 4.15.

**Exercise 51.** Show that the bound of Proposition 4.18 cannot be improved.

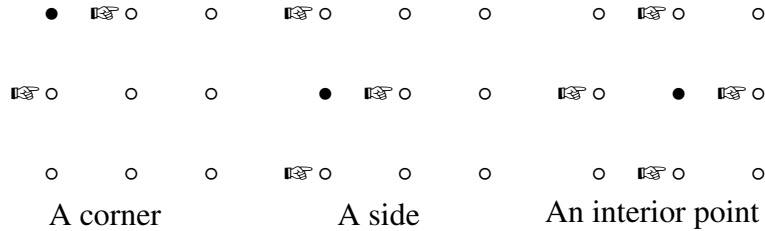
**Exercise 52.** Show that  $c \otimes^Y c = c$  and that  $c \otimes^{YZ} c = c \otimes^{Z,Y} c$ . Show that  $(c \otimes^{YZ} c) = c$  does not generally hold.

**Exercise 53.** Show that  $c \otimes^Y d = d \otimes^Y c$ . Give an example to show that in general  $c \otimes^{YZ} d = d \otimes^{YZ} c$  is false.

## 4.5 Concepts and Pictures

Up to now it looked as if concepts were a complicated sort of relations. However, the intention is that in reality things are the other way around: that relations are a complicated sort of concept. In this section I'd like to sketch a very different approach to concepts using *pictures*; moreover, I shall show that concepts are not at all difficult to use. The approach is just one among many, and only illustrates

Figure 4.1: Types of Points



the way things might go. We shall assume throughout that basic relations are *symmetric* so that questions of ordering between the argument places is irrelevant.

We want to define all sentence meanings as certain sets of pictures; a picture in turn is an array of coloured dots. Hence we construe pictures as functions from arrays into the set of colours. A simple approach would be to say that an array is a certain subset of, say,  $\mathbb{N}^2$  (if the picture is planar) or  $\mathbb{N}^3$  (for spatial pictures). However, we prefer a slightly more abstract definition. We start with a set  $L$ , the set of **locations**. A **space** is a pair  $\mathcal{S} = \langle L, A \rangle$  where  $A \subseteq \binom{L}{2}$  is a relation, the **adjacency relation**. Here,  $\binom{L}{2}$  is the set of 2-element subsets of  $L$ . We define  $L^+$  to be the transitive closure of  $L$ . (It follows that  $L^+$  is symmetric and reflexive (if  $\text{card } L > 1$ ).) We assume that any two points are related via  $L^+$ . This means that  $\mathcal{S}$  is **connected**.

Let us assume that  $L$  is a subset of  $\mathbb{N}^2$ , and that  $\{(x_0, x_1), (y_0, y_1)\} \in A$  iff  $|x_1 - x_0| + |y_1 - y_0| = 1$ . This means that either (a)  $x_1 = x_0$  and  $y_1 = y_0 \pm 1$ , or (b)  $y_1 = y_0$  and  $x_1 = x_0 \pm 1$ . Say that  $\ell'$  is a **neighbour** of  $\ell$  if  $\{\ell', \ell\} \in A$ . It follows that any  $\ell \in L$  has at most 4 neighbours. We shall assume that no points have exactly zero or one neighbour; this excludes some trivial sets. From this we can define three sets of points (see Figure 4.1):

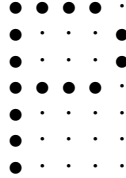
1. **Corners:** have exactly two neighbours;
2. **Sides:** have exactly three neighbours;
3. **Interior points:** have exactly four neighbours.

If  $\ell$  is interior, let  $n_0, n_1, n_2$  and  $n_3$  be its neighbours. We say that  $n_1$  is **across** from  $n_0$  if there is exactly one  $p$  such that (1)  $\{n_0, p\}, \{p, n_1\} \in A$  and (2)  $p$  is not a corner.. There is a exactly one point which is across from  $n_0$ ; let  $n_1$  be across from  $n_0$  and  $n_3$  across from  $n_2$ . In the space  $\mathcal{S}$ , the relation of acrossness can be used to define lines: a **line** is a maximal connected subset  $G \subseteq L$  such that for any three points  $p, q, r$  such that  $\{p, q\}, \{q, r\} \in A$ , then  $p$  is across from  $r$ . It is easy to see that if  $p$  and  $q$  are neighbours, there is a unique line that contains them. In the plane  $\mathbb{N} \times \mathbb{N}$ , lines are subsets that are either horizontal or vertical. the vertical lines are parallel to each other, so are the horizontal lines. So we say that two lines  $G$  and  $G'$  are **parallel** if  $G \cap G' = \emptyset$ . If  $G$  and  $G'$  are not parallel, then we require that  $\text{card}(G \cap G') = 1$ . In the plane, if  $G$  and  $G'$  are parallel and  $H$  is not parallel to  $G$ , then it is also not parallel to  $G'$ . Now pick any line  $H$  and let  $\mathcal{H} := \{H' : H \cap H' = \emptyset\}$ . This defines the set of horizontal lines; pick another line  $V$  and put  $\mathcal{V} := \{H : H \cap V = \emptyset\}$ . This defines the set of vertical lines. Any line is either of these sets.

I stress that there is no way to say in advance which line is horizontal; the map  $\bowtie: (x, y) \mapsto (y, x)$  maps  $L$  onto some different set  $L^\bowtie$  preserving the adjacency but interchanging ‘horizontal’ and ‘vertical’. Furthermore, by symmetry of  $A$ , the directions ‘up’ and ‘down’ cannot be distinguished; likewise, the directions ‘left’ and ‘right’. To fix them, we need to introduce extra structure. A **coordinate frame** is a triple  $\mathcal{C} = \langle o, r, u \rangle$  in  $L$  such that  $\{o, r\}, \{o, u\} \in A$ , and  $u$  is not across from  $r$ . The line containing  $o$  and  $r$  defines  $\mathcal{H}$ , and the line containing  $o$  and  $u$  defines the set  $\mathcal{V}$ . Now pick any point  $p$ . It has four neighbours,  $q_0$  through  $q_3$ . Which one of them is ‘up’ from  $p$ ? First, there is exactly one line in  $\mathcal{V}$  through  $p$ , and it contains, say  $q_0$ . It also contains one more neighbour of  $p$ , say  $p_1$ . Then either  $q_0$  is ‘up’ from  $p$  or  $q_1$  is. To settle the problem which is which we need to introduce more notions. First, the **distance**  $d(x, y)$  between  $x$  and  $y$  is  $n$  if  $n$  is the smallest number such that there is a sequence  $\langle x_i : i < n + 1 \rangle$  with  $x_0 = x$ ,  $x_n = y$  and for all  $i < n$   $\{x_i, x_{i+1}\} \in A$ .  $p$  is **between**  $q$  and  $r$ , in symbols  $B(r, p, q)$  if  $p, q$  and  $r$  are on a line, and  $d(r, p), d(r, q) < d(p, q)$ . Using betweenness it is finally possible to define what it is for two pairs  $(p, q)$  and  $(p', q')$  to be oriented the same way. This is left as an exercise. It follows that  $q_0$  is up from  $p$  iff  $(p, q_0)$  is equioriented with  $(o, u)$ .

Pictures are pairs  $\langle \mathcal{S}, f \rangle$ , where  $f : L \rightarrow C$  is a function assigning each location a colour. For simplicity we assume we have just two colours: black and white. Black represents point of matter; white points represent ‘air’. In that case,

Figure 4.2: Pictures by Pixels



in place of  $f$  we may just name the set of black points. This is a well known type of representations. For example, printers prints pictures by means of little dots of ink placed at certain points of a grid. Letters can be sufficiently clearly represented using a 5 by 7 grid (see Figure 4.2). Thus we represent ‘matter’ with a subset of the locations. A **picture** is a pair  $\mathcal{P} = \langle \mathcal{S}, B \rangle$  where  $\mathcal{S}$  is a space and  $B \subseteq L$ . We shall continue to assume that  $\mathcal{S}$  is a rectangular subset of  $\mathbb{N} \times \mathbb{N}$ . An **object** in  $\mathcal{P}$  is a maximally connected subset of  $B$ . Here,  $C \subseteq B$  is connected if for any two points  $p, q \in C$  we have  $\{p, q\} \in (A \cap \binom{B}{2})^+$ . (In plain words: there is a sequence of pairwise adjacent points inside  $B$ .)  $\mathcal{O}(\mathcal{S})$  is the set of objects of  $\mathcal{S}$ . Objects are therefore defined through their location. An **object schema** is a picture  $\mathcal{P} = \langle \langle P, N \rangle, C \rangle$  containing a single object. We may for simplicity assume that the picture is a minimal rectangle around the object. Then we may say that  $\mathcal{S}$  **contains** an object of type  $\mathcal{P}$  if there is a function  $f : P \rightarrow L$  such that (a)  $\{x, y\} \in N$  iff  $\{f(x), f(y)\} \in A$ , and (b)  $f[C]$  is an object of  $\mathcal{S}$ . The function  $f$  is also said to be a **realisation** of  $\mathcal{P}$  in  $\mathcal{S}$ . The same object of  $\mathcal{S}$  can realise an object schema in different ways. This is exactly the case if it possesses internal symmetry.

Obviously, this is the most simple of all scenarios. We define an object schema as an arrangement of pixels and then declare any pixel schema that has identical arrangements (up to flipping it upside down or left-to-right) as an instantiation of that object schema. Evidently, however, we may easily complicate the matter by allowing more fancy embeddings: those that keep distances ratios intact (thus allowing to shrink or magnify the picture) or those that rotate the picture. This makes full sense only if pictures are defined over the real plane, but nothing essential hinges on that, except that there is no more adjacency relation and we have to work with topology and the metric. Let us remain with the scenario as

developed so far. It then is quite easy to see how object schemata can be learnt. We need to be shown a single instance. Properties of objects (the denotations of common nouns) are inferred from their instances. It is not our concern to see how that can be done; this is the domain of cognitive science. Basically, it is done by inferring a set from some of its members (for example by constructing so-called Voronoi cells, see [Gärdenfors, 2004]).

The way such learning can take place in language is as follows. Let Paul be our language learner. Paul is shown a picture containing a single object, say a football, and is told that it is a ball. Thus, Paul will get the following data:

(4.56)  $\langle /This\ is\ a\ ball./, \boxed{\text{⚽}} \rangle$

To the left we have an utterance, to the right a picture. That the utterance is paired with a specific picture is of some importance. Now, Paul will have to do some inference here to arrive at the fact that /ball/ denotes the object schema  $\boxed{\text{⚽}}$  rather than the picture. Once this achieved, however, he is able to identify the concept denoted by /ball/. In a similar fashion he will learn other unary concepts such as “flag”, “hut”, “tent”, “telephone”, and so on.

The next step from here is to learn the meaning of relational concepts. Let us take the concept “to the left of”. Unlike the denotation of common nouns, it is not identifiable by means of a single picture, since it is a relation between objects. How then can it be learned? The answer is that it is learned in basically the same way. Paul is presented with a picture and a sentence

(4.57)  $\langle /The\ scissor\ is\ to\ the\ left\ of\ the\ ball./, \boxed{\text{✂ ⚽}} \rangle$

This picture allows to establish an association between the phrase /the scissor/ and the object to the left (since it is the only scissor) and between the phrase /the ball/ and the object to the right. This requires only knowledge of the meaning of the expressions. Similarly, Paul will encounter the following pair:

(4.58)  $\langle /The\ square\ is\ to\ the\ left\ of\ the\ heart./, \boxed{\text{□ ♥}} \rangle$

He may come to realise that the concept “left of” is independent of the shape and size of the objects involved, and that it is about the location of the objects with respect to each other. In that case it can be represented just like an object schema, using a set of pictures. The burden is then entirely on the kinds of maps (“deformations”) that one is allowed to use to embed such pictures in others. It is



not our main concern to do that; rather we wish to point out that the learning of the concept “left of” is no more complex using concepts than it is using relations.

How then is “right of” learnt? Basically the same way. It could be using the following data.

(4.59)  $\langle /The\ ball\ is\ to\ the\ right\ of\ the\ scissor./, \boxed{\text{✂} < \text{⊗}} \rangle$

Here we can appreciate for the first time that concepts really *are* simpler. The picture shown is namely absolutely the same. However, in conventional representations we would write (4.59) as

(4.60)  $right(\iota x.ball(x), \iota x.scissor(x))$

By contrast, the sentences of (4.57) would be rendered as

(4.61)  $left(\iota x.ball(x), \iota x.scissor(x))$

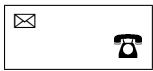
The two formulae are not the same. The positional regime in the formulae forbids us from treating them the same. To get an identical encoding we need to translate “right” systematically as “left” and invert the linking. This is what the concepts do anyway. Paul will learn that whatever is to the left of /right/ will be on the right in the picture of what is to the right of /right/. I should point out that it is certainly not necessary that the meaning of (4.57) need not exactly be the same as (4.59). In that case /right/ denotes a different concept than /left/. We shall worry no further about that possibility. It should however be said that there can be concomitant differences in the choice between (4.57) and (4.59) stemming from different sources. I mention here that constructions of the form “X is in location Y” generally indicate that Y is more stable, less movable, or bigger than X (see [Talmy, 2000]).

(4.62) The bicycle is to the left of the house.

(4.63) ?The house is to the right of the bicycle.

Again, this issue seems to be orthogonal to the one at hand. (Notice also that (4.63) is not *false*, just inappropriate.)

I shall now test Paul’s knowledge of English. We give him the picture (4.64)

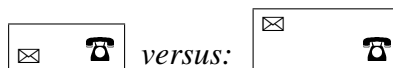
(4.64) 

and ask him:

(4.65) Is the letter to the left of the phone?

Paul will perform the following steps:

- ① Compare the two arguments of /left/ in (4.65) in (4.57). The comparison on the basis of form alone yields that /the scissor/ must be associated with /the letter/ and /the ball/ with /the phone/.
- ② Take the picture of (4.57) and do the following replacement: replace the scissor by the letter and the ball by the phone.
- ③ Compare the resulting picture with the one given:



- ④ If there is an admissible deformation to take us from left to right for the concept “left” then the answer is “yes”.

Thus, the entire burden is still in learning the correct meaning of the geometry of “left”. Learning the associations with syntactic arguments is very easy by comparison. Moreover, a semantics based on relations offers no advantage.

We have deliberately picked the concept “left”. Unlike concepts denoted by verbs, geometric notions do not allow to pick out one of the arguments by means of intrinsic properties. For example, the sentence “John is pushing the cart.” is true because it is John who exerts force on the cart and not conversely. Likewise, it is known that directionals modify the mover in an event and no other constituent. Thus “John threw the banana out of the window.” means that John threw the banana and *it* went out of the window. If John decides to jump out of the window, while tossing the banana onto the kitchen table, that does not make the sentence true. The mechanism for learning such concepts is essentially the same. However, while the linking in relational nouns and adjectives has to be learned on a case by case basis, the linking on verbs sometimes allows to make big abstractions. This just means that the knowledge of how linking is to be effected becomes more abstract.

Let us finally turn to another complication, namely passive, or relation change in general.

(4.66) John throws the banana out of the window.

(4.67) The banana is thrown out of the window.

It is obvious that a correct learning of English will consist in realising that there are different verb forms, namely active and passive, and that what they signal is that the linking has to be different in the two cases. From this point on there are two choices: Paul might start acquiring *two* different linkings for the verbs, one active and one passive; or Paul might develop a recipe of deriving the linking in the passive sentences from the linking in active sentences. How he goes about is to a large degree a question of how the languages are structured (in other words: how systematic the active passive change really is).

I close this section with a few remarks about what we have done. We have described sentences as properties of pictures. There was therefore only one entity in semantics: that of a picture. To describe how it is that we arrive at the interpretation of a sentence, however, we complicated the ontology. If a sentence has subjects, something must correspond to them. Thus we introduced individuals, concepts, and so on into the semantics. However, ontologically these were considered derived objects. I constructed a function that will derive from a picture  $\mathcal{P}$  the set of its objects  $\mathcal{O}(\mathcal{P})$ . The next object we introduced are the object schemes; an object scheme  $P$  is a picture  $\mathcal{Q}$  together with a family  $F$  of admissible embeddings. An object  $o \in \mathcal{O}(\mathcal{P})$  has a property  $P$  if there is an admissible embedding  $f : \mathcal{Q} \rightarrow \mathcal{P}$  that such that the image of the black points is exactly  $o$ .

**Exercise 54.** Define the relation of “having same orientation” using betweenness in a plane. *Hint.* Start by defining it for pairs of points on the same line. Then show it can be projected to other, parallel lines.

## 4.6 Ambiguity and Identity

We have shown earlier that sentences are ambiguous, and they can be so either because the words have several meanings or because a given exponent has several derivations. In view of this ambiguity we must reassess our notion of what it is

that makes a sentence true. Under the standard definitions in logic we declare a sentence true if it denotes the value 1 or the true concept, whichever. However, if a sentence is ambiguous this creates a difficulty. Consider the word /crane/. It has two meanings: it denotes a kind of birds, and a kind of machine. This means that the lexicon contains two signs, where  $\text{crane}_1$  is the concept of cranes (a type of bird) and  $\text{cranes}_2$  is the concept of cranes (a kind of machine).

(4.68)  $\text{BCR} := \langle \text{crane}, \text{crane}_1 \rangle$

(4.69)  $\text{MCR} := \langle \text{crane}, \text{crane}_2 \rangle$

Consider now the following sentence.

(4.70) Cranes weigh several tons.

This sentence has two derivations. Unless partiality strikes, in a structure term containing BCR we can replace BCR by MCR, and the new term unfolds to a sign with the same exponent (but different meaning).

(4.70) is false if we interpret /cranes/ as talking about birds (that is, if we take the structure terms to contain BCR rather than MCR), but true in the other understanding of the word. It is the converse with

(4.71) Cranes can fly.

This creates a tension between the notion of ‘true under an understanding’ and ‘true simpliciter’. We shall propose (not uncontroversially) that a sentence is true simpliciter if it has a structure term under which it is true. This is a matter of convention, but for the cases at hand not far off the mark. It then is the case that both (4.70) and (4.71) are true.

Now what about negated sentences? Here we must distinguish between two kinds of negations. There is an inner negation and an outer negation. The inner negation produces a negated sentence, while the outer negation denies the truth of the sentence. Let’s look at negation formed by /it is not the case that/.

(4.72) It is not the case that cranes weigh several tons.

If taken as outer negation, this sentence is false (because (4.70) is true). If taken as inner negation, it is true. To see this, let us imagine that we do not have the word

/cranes/, but in fact two: /cranes<sub>1</sub>/, denoting the birds, and /cranes<sub>2</sub>/, denoting a kind of machines. Then (4.70) is true if either of the following sentences is true:

(4.73) Cranes<sub>1</sub> weigh several tons.

(4.74) Cranes<sub>2</sub> weigh several tons.

(4.70) is false if *both* (4.73) and (4.74) are false. It is possible through to negate both of them individually:

(4.75) It is not the case that cranes<sub>1</sub> weigh several tons.

(4.76) It is not the case that cranes<sub>2</sub> weigh several tons.

The first is true while the second is false. In English, where the two concepts are denoted by the same word, (4.75) and (4.76) are both expressed by (4.72). Since (4.76) is true, so is therefore (4.72).

I should say, however, that the notion of outer negation cannot be implemented in the present system without major changes. For if outer negation is a sign in its own right, its meaning is a quantifier over structure terms. Semantically this is not possible to implement. It is not clear to me whether or not outer negation can be expressed in embedded sentences. If it cannot be expressed, the present theory can obviously be adapted rather straightforwardly; but if it can be expressed, then the adaptations are indeed major. They would require namely a grammar that uses the language transform  $L^{\S}$  of  $L$  rather than  $L$  itself (see Page 123 for a discussion of  $L^{\S}$ ).

The previous can be used to shed light on identity statements as well. Consider the sentence

(4.77) The morning star is the evening star.

It is true if and only if the star that is the morning star is the same star as the evening star. It happens to be the case that they actually *are* the same. If John however is unaware of this, then he believes that (4.77) is false and that (4.78) is true.

(4.78) The morning star is the evening star.

This problem has been extensively dealt with in philosophy. We shall not go into that discussion. Rather, we shall discuss how our definitions change the way in which this puzzle must be discussed.

**Example 61.** Let  $M = \{x\}$ . Furthermore, we shall assume that our language has the following basic signs.

$$(4.79) \quad \mathcal{J}(f_0) := \langle \text{the morning star}, \{x\} \rangle$$

$$(4.80) \quad \mathcal{J}(f_1) := \langle \text{the evening star}, \{x\} \rangle$$

And let it have one mode:

$$(4.81) \quad \mathcal{J}(f_2)(\langle e_0, m_0 \rangle, \langle e_1, m_1 \rangle) := \langle e_0 \hat{\text{is}} e_1, m_0 \otimes m_1 \rangle$$

Here,  $\otimes$  is defined as intersection of two 1-concepts by intersecting their minimal members. Let  $L_1$  be the language defined by all definite terms. It is

$$(4.82) \quad L_1 := \{ \langle \text{the morning star}, (\{x\})_M \rangle, \langle \text{the evening star}, (\{x\})_M \rangle, \\ \langle \text{the morning star is the morning star}, 1 \rangle, \\ \langle \text{the morning star is the evening star}, 1 \rangle, \\ \langle \text{the evening star is the morning star}, 1 \rangle, \\ \langle \text{the evening star is the evening star}, 1 \rangle \}$$

Now let  $N = \{v, w\}$ . We assume the same signature, but instead the following interpretation:

$$(4.83) \quad \mathcal{K}(f_0) := \langle \text{the morning star}, \{v\} \rangle$$

$$(4.84) \quad \mathcal{K}(f_1) := \langle \text{the evening star}, \{w\} \rangle$$

$$(4.85) \quad \mathcal{J}(f_2)(\langle e_0, m_0 \rangle, \langle e_1, m_1 \rangle) := \langle e_0 \hat{\text{is}} e_1, m_0 \otimes m_1 \rangle$$

Let  $L_2$  be the language defined by this interpretation. Then

$$(4.86) \quad L_2 := \{ \langle \text{the morning star}, (\{v\})_M \rangle, \langle \text{the evening star}, (\{w\})_M \rangle, \\ \langle \text{the morning star is the morning star}, 1 \rangle, \\ \langle \text{the morning star is the evening star}, 0 \rangle, \\ \langle \text{the evening star is the morning star}, 0 \rangle, \\ \langle \text{the evening star is the evening star}, 1 \rangle \}$$

We have the following result: there are *two* languages, not one, whose corresponding string language is the same, and we even have two string identical grammars. But nevertheless, qua interpreted languages,  $L_1$  and  $L_2$  are different.  $\odot$

The example has the following moral. Two languages cannot be the same if the models are not the same. Thus, to say that John and Paul speak the same language—in the sense of interpreted language, which we take to be the default—requires that their interpretations are the same. If Paul is convinced that the morning star is the evening star and John thinks they are different then Paul and John do not speak the same language. In order for them to speak the same language we require that not only the expressions are the same, we also require that the expressions have the same meaning. And ‘same’ must be taken in a strict sense: both John and Paul would be required to take the expressions /the morning star/ to denote the same thing, and likewise /the evening star/. But they do not. There are in fact two reasons why two people can fail to share the same language. One is as just described: they disagree on the truth value of some sentences. Another more subtle case is described in the next example.

**Example 62.**  $L_3$  is like  $L_1$  except that  $y$  takes the place of  $x$ . Thus, for example,

$$(4.87) \quad L_3 := \{ \langle \text{the morning star}, (\{y\})_M \rangle, \langle \text{the evening star}, (\{y\})_M \rangle, \\ \langle \text{the morning star is the morning star}, 1 \rangle, \\ \langle \text{the morning star is the evening star}, 1 \rangle, \\ \langle \text{the evening star is the morning star}, 1 \rangle, \\ \langle \text{the evening star is the evening star}, 1 \rangle \}$$

Now let  $P = \{y\}$ . We assume the same signature, but instead the following interpretation:

$$(4.88) \quad \mathcal{L}(f_0) := \langle \text{the morning star}, \{y\} \rangle$$

$$(4.89) \quad \mathcal{L}(f_1) := \langle \text{the evening star}, \{y\} \rangle$$

$$(4.90) \quad \mathcal{L}(f_2)(\langle e_0, m_0 \rangle, \langle e_1, m_1 \rangle) := \langle e_0 \hat{=} \text{is} \hat{=} e_1, m_0 \otimes m_1 \rangle$$

The grammars  $\langle \Omega, \mathcal{J} \rangle$  and  $\langle \Omega, \mathcal{L} \rangle$  are naturally equivalent. ⊛

The languages  $L_1$  and  $L_3$  are different, yet in an abstract sense identical. Now picture the case where George speaks  $L_3$ . We would like to say that George and Paul speak the same language, but we cannot. In fact, this is as it should be. For notice that we must distinguish (for natural language) two notions of language. There is a private language, where expressions are interpreted as objects or constructs in a speaker; and a public language where expressions are interpreted with

real objects (if applicable). We think for example that the public meaning of /the morning star/ is Venus, as is the public meaning of /the evening star/. The private language of an individual speaker needs to be ‘connected’ to the public language in the correct way. This is similar in the distinction between phonemes and sounds. While two speakers can share the same phonemic system it may turn out that the two systems are differently realized in terms of sounds. And likewise it may happen that while Paul thinks that the morning star is the evening star and both happen to be Venus, it may also happen that George thinks that the morning star and the evening star are Mars. The private languages of Paul and George are different for the trivial reason that the internal objects of both Paul and George must be different; but we can easily establish a correspondence between them, an isomorphism, that makes them the same. And so the private languages of Paul and George are the same up to isomorphism, yet their public languages are different. The puzzle is thus resolved by appeal to different *de lingua* beliefs, to use a phrase of [Fiengo and May, 2006]. The idea of [Fiengo and May, 2006] is roughly that what is behind many puzzles of identity is that speakers hold different beliefs concerning the referent of expressions. In the theory proposed here, this is cashed out as follows. The **abstract language** is language where meanings are identifiable up to equivalence (as established in Section 3.7). Any two speakers can speak the same abstract language, so the abstract language is not the private language. Neither is it the public language. For that, we also need to **ground** a language by providing translations into real world objects. Abstract language behaviour can be established using logical connections between sentences, while concrete language behaviour can be established by asking people about meanings in terms of observable facts.<sup>4</sup> This is just a sketch of a solution but it serves as a formal explication of [Fiengo and May, 2006] who actually think that many sentences also express what they call a *de lingua belief*. A *de lingua belief* is a belief about what expressions denote. If the present is correct, it is a belief about the public language.

One puzzle that Fiengo and May discuss at length is the Paderewski puzzle by Kripke. It goes roughly as follows. Max goes to a concert by a musician named Paderewski and comes to believe that he is a great musician. Later he visits a political rally by a person named Paderewski. He comes to think that the latter

---

<sup>4</sup>This is evidently a simplified scenario. The visible facts may not be the same across speakers, thus accounting for a different layer of confusion. But it is important to note that the distinction between what is abstract in a language and what is not is real. In a sense, the fact that Tully is Cicero is not part of the abstract language.



person is actually a bad musician. So he holds two beliefs.

(4.91) Paderewski is a great musician.

(4.92) Paderewski is a bad musician.

It so turns out that the two people are one and the same. The philosophical problems arise from the fact that under certain views of reference Max holds inconsistent beliefs. Both [Fine, 2007] and [Fiengo and May, 2006] discuss this problem. Again we need not go into the philosophical detail here. What interests us is what may linguistically be said to go on. The idea is that for Pawel, who knows (or believes) that both persons are the same, ‘Paderewski’ is unambiguous. For Max it is not. So, the language of Max has two signs, say,  $\langle \text{Paderewski}, \{x\} \rangle$  and  $\langle \text{Paderewski}, \{y\} \rangle$ , while the language of Pawel only has one such sign, say  $\langle \text{Paderewski}, \{v\} \rangle$ . Thus, for Max the expression /Paderewski/ is ambiguous, for Pawel it is not. Given our notion of truth for ambiguous sentence, it is correct for Max to hold both (4.91) and (4.92) true. There is no logical problem, since the sentence is simply ambiguous. This contrasts with the idea of [Fiengo and May, 2006] who think that names are not expressions. They can only occur in the form  $[_1 \text{Paderewski}]$ , where the brackets are used to keep track of different objects. In the theory proposed here there is no sense in disambiguation on the syntactic level. This must be done in the semantics. Consequently, also the two occurrences of the name in the sentence

(4.93) Paderewski is Paderewski.

cannot simply be told apart by indexation so that one can distinguish between, for example,

(4.94) Paderewski<sub>1</sub> is Paderewski<sub>1</sub>.

(4.95) Paderewski<sub>1</sub> is Paderewski<sub>2</sub>.

The reason, again, is that there is no surface indication of such a difference. Instead, in order to be clear, Max must use some expression that make the referent unique. Notice that Max also agrees to the (inner!) negation of (4.93):

(4.96) Paderewski is not Paderewski.

The difference between this approach and [Fiengo and May, 2006] is brought out also by the way in which Pawel can make Max aware that he is wrong about

Paderewski. For it is not enough for him to point out (4.96), for that is what is also true for Max. Rather he must use a sentence that would not be true, for example

(4.97) There is only one Paderewski.

The problem is that Pawel cannot make himself understood to Max by using the name simpliciter. He must in order to discriminate his beliefs from Max's beliefs use sentences that come out differently. What [Fiengo and May, 2006] have in mind is that Pawel can also use a certain version of (4.93), for example

(4.98) But Max, Paderewski *IS* Paderewski.

But again, how is Max to interpret this if he cannot see which of the Paderewskis is pointed to on each of the occasions?

**Exercise 55.** In Example 61 the word /is/ is syncategorematic. Show that this syncategorematic use can be eliminated from the grammar.

## 4.7 Profiling

As I have indicated at many places there is a difference between what is commonly referred to as modeltheoretic semantics and the more popular representational semantics. It has not always been openly admitted by semanticists that the representations involved in many brands of formal semantics do not use meanings in the sense of truth conditions but that they rather are just pieces of notation. Such is the case with DRT, minimal recursion semantics, semantics used in connection with TAGs, underspecification semantics, continuations, and so on. If meanings only contain truth conditions, then all these semantics could not ever claim to implement a compositional approach to meaning. However, such argumentation misses a point. For one line of defense is still open and should be considered: that it is not the only objective to account for truth conditional meanings but rather *also* for internal meanings. Thus I believe that the justification for using such representations cannot be found in the truth conditions that they formulate. Rather, it must be in the fact that these objects are essentially what humans use. This is an empirical question and will have to be left to empirical research. However, I shall just add a few remarks about the necessity of considering internal meanings. If

we take, for example, the notion of a dog to be the set of all dogs, then that object is not of the kind we can have in our head. We may say instead that the meaning is a particular algorithm (for recognising dogs); but even that has a similar consequence. The algorithm turns out to be abstract, too. The particular procedure that one person uses to differentiate dogs from other animals might be different from that of some other person in certain insignificant ways. We will then still say that the two people have the same algorithm, though their implementation, that is, the concrete procedures, are different.

The crucial fact about the concreteness of meanings is that to understand whether or not two concrete meanings  $m$  and  $m'$  instantiate the same abstract meaning must be decided by explicit manipulation of the representations. This is the same in logic, where we distinguish between two formulae representing the same truth condition. Since truth conditions are too big to be stored directly we rely instead on a calculus that manipulates representations up to truth conditional equivalence. This picture undermines much of what I have said so far about semantics since it moves us away from a static notion of meaning and towards a dynamic semantics based on reasoning whose objects are symbolic in nature. I shall not continue that line since it is too early to tell how such an account may go.

It so turns out, however, that human languages are still different. There are certain things that have been argued to exist in internal representations for which there is no obvious external correlate. One such thing is profiling. Profiling is the way in which objects in an array are distinguished from each other, by making one more prominent than the others. We can explain the difference between “left” and “right”, for example, in terms of profiling. While they both denote the same concept, the profile of “left” is inverse of that of “right”. How can this be understood? In the pictures we can simply add a pointer to the profiled entity (in cognitive grammar, prominent elements are drawn using thick lines). If we denote concepts by formulae then we can use underlining to do the same: thus,  $\langle \text{left}'(x, \underline{y}) \rangle$  and  $\langle \text{left}'(\underline{x}, y) \rangle$  are concepts in which different elements are profiled. If we use concepts, we reserve, say, the first column for the profiled element and restrict permutation in such a way that it does not permute the first column with any other. There is a temptation to think of profiling as just another instance of sort. But we have to strictly distinguish the two. The two objects involved in the relation “left” (and “right”) are not sortally distinct. Moreover, one and the same object can at the same time be to the left of an object, and to the right of another.

This cannot happen if a different profile means different sort. However, from the standpoint of combining meanings profiling has the same effect, namely to reduce the possibilities of combining two concepts.

In the first part of this section I shall outline a formalism for such meanings. In the second half I show how this gets used in practice.

Let  $S$  be a set of sorts. So far we have construed concepts as sets of relations. The minimal members of a relation had to be of similar type. Now we think of the relations of a concept to be divided into subparts, each corresponding to a particular profile. We allow individual sorts to be profiled independently.

**Definition 4.20** *Let  $P$  be a set of profiles and  $M$  a set. A  $P$ -profiled relation over  $M$  is a pair  $\mathcal{R} = \langle \vec{p}, R \rangle$  where  $R$  is a relation and  $\vec{p} \in P^*$  of length identical to the length of  $R$ .*

The relation  $R$  contains vectors  $\langle x_0, x_1, \dots, x_{n-1} \rangle$ . When paired with the sequence  $\langle p_0, p_1, \dots, p_{n-1} \rangle$  this means that  $x_i$  will have the profile  $p_i$ . Since the profile is paired with the entire relation, the profile  $p_i$  is also given to  $y_i$  in  $\langle y_0, y_1, \dots, y_{n-1} \rangle \in R$  in  $\mathcal{R}$ . One may or may not want to impose requirements on the profiling. For example, suppose there is a label saying that the element is in focus; this label we do not want to be distributed to more than one column. But such requirements can always be added later.

A profiled concept is a set of profiled relations. We details are similar to those of Section 4.3. The profiled concept generated by  $\mathcal{R}$ , also written  $\llbracket \mathcal{R} \rrbracket_{\mathcal{M}}$ , is the least set closed both ways under the following operations.

- ①  $\pi[\langle \vec{p}, R \rangle] := \langle \pi(\vec{p}), \pi[R] \rangle$ ,  $\pi$  a permutation of the set  $|\vec{p}| = \{0, 1, \dots, |\vec{p}| - 1\}$ ;
- ②  $E_{s,q}(\langle \vec{p}, R \rangle) := \langle \vec{p} \cdot q, R \times M_s \rangle$ ;
- ③  $D_i(\langle \vec{p}, R \rangle) := \langle \vec{p} \cdot p_i, \{ \vec{x} \cdot x_i : \vec{x} \in R \} \rangle$ .

Notice that when duplicating a column we must also duplicate the corresponding profile. It is therefore quite possible to have two identical columns, as long as they have different profiles. Notice that full columns are discarded regardless of their profile.

The **deprofiling** of  $\langle \vec{p}, R \rangle$ ,  $\delta(\langle \vec{p}, R \rangle)$ , is simply  $R$ . Similarly, we define the deprofiling of a profiled concept.

$$(4.99) \quad \delta(\llbracket \mathcal{R} \rrbracket) := \{S : \text{there is } \vec{q}: \langle \vec{q}, R \rangle \in \llbracket \mathcal{R} \rrbracket\}$$

So,  $\delta(\mathcal{C}) = \delta[\mathcal{C}]$ . The following gives the justification for this definition. Its proof is left as an exercise.

**Proposition 4.21**  $\delta(\llbracket \mathcal{R} \rrbracket)$  is a concept.

There is a converse operation of introducing a profiling. While we could do that on a concept-by-concept basis, there are more interesting methods.

**Definition 4.22** Let  $Y$  be a linking aspect and  $f : \mathbb{N} \rightarrow P$  a function. Then define the profiled concept  $f^Y(c)$  as follows.

$$(4.100) \quad f^Y(c) := \llbracket \langle f \upharpoonright \text{card}(Y(c)), Y(c) \rangle \rrbracket$$

In this definition, assume that  $\text{card}(Y(c)) = n$ . Then  $f \upharpoonright \text{card}(Y(c))$  is the restriction of  $f$  to  $n = \{0, \dots, n-1\}$ . This is then viewed as the sequence  $\langle f(0), f(1), \dots, f(n-1) \rangle$ . The idea is that all we need to specify is the way in which the positions are profiled; the rest is done by the linking aspect, which lines up the columns of the relation in a particular way.

The crucial difference between profiled concepts and ordinary concepts is that we can use the profiles to define the linking; and that we can also change the profile if necessary (unlike the typing). In principle, since the profiling is arbitrary, we consider two profiled concepts as basically identical if they have the same profile.

**Definition 4.23** Two profiled concepts  $\mathcal{C}$  and  $\mathcal{D}$  are said to be **homologous** if  $\delta(\mathcal{C}) = \delta(\mathcal{D})$ .

Any change from a profiled concept to a homologous profiled concept is thus considered legitimate. There various methods to define such a change for the entire space of concepts. Here is one.

**Definition 4.24** Let  $S$  be a set of sorts and  $P$  a set of profiles. A **reprofiling** is a family  $\{\rho_s : s \in S\}$  of maps  $\rho_s : P \rightarrow P$ . The reprofiling of a profiled relation  $\langle \vec{p}, R \rangle$  of type  $\vec{s}$  is the relation  $\rho(\mathcal{R}) := \langle \rho_R\{\vec{p}\}, R \rangle$  which is defined as follows.

$$(4.101) \quad \begin{aligned} \rho_R\{p_i\} &:= \rho_{s_i}(p_i) \\ \rho_R\{\vec{p}\} &:= \langle \rho_R\{p_i\} : i < \text{card}(R) \rangle \end{aligned}$$

Notice that the type of the relation is recoverable from the relation itself (in contrast to its profile). So the reprofiling assigns to elements of type  $s$  and profile  $p$  the new profile  $\rho_s(p)$ , whereas the type remains the same.

**Proposition 4.25** Let  $\mathcal{C}$  be a profiled concept and  $\rho = \{\rho_s : s \in S\}$  a reprofiling. Then  $\rho[\mathcal{C}]$  is a profiled concept.

Again the proof is straightforward.

The simplification introduced by profiling is considerable. Suppose for example we want to conjoin two concepts. Then we can only do this if we have a linking aspect. However, linking aspects are not the kind of object that is finitely specifiable. Thus, unlike syntactic rules, the semantic combination rules based on concepts are arbitrarily complex. In Section 5.3 I shall give an example of a grammar for a fragment of English that essentially uses linking aspects only for the basic entries of the lexicon. If one wants to treat language in its full complexity one will be forced to do either of two things: make the linking aspect dynamic, that is, to be computed on the side; or introduce profiling. In this section I shall explore the second option.

Now that we have profiled concepts we may actually take advantage of the profiling in defining combinations of concepts. Our example here concerns the definition of linking aspects.

**Example 63.** Arbitrarily embedded relative clauses.

$$(4.102) \quad \begin{array}{c} \text{a dog that saw a cat that chased a mouse that ate} \\ \text{a cheese} \end{array}$$

Let  $D = \{d_i, c_i, m_i, h_i : i \in \mathbb{N}\}$  be the domain. There is only one sort. Let us define three binary relations:

$$(4.103) \quad \begin{aligned} E &:= \{\langle m_0, h_0 \rangle\} \cup \{\langle d_i, h_{i+1} \rangle : i \in \mathbb{N}\} \\ C &:= \{\langle c_i, m_i \rangle : i \in \mathbb{N}\} \cup \{\langle c_i, d_{i+1} \rangle : i \in \mathbb{N}\} \\ S &:= \{\langle d_i, c_i \rangle : i \in \mathbb{N}\} \cup \{\langle m_i, d_{2i} \rangle : i \in \mathbb{N}\} \end{aligned}$$

$$(4.104) \quad \begin{aligned} \mathcal{J}(g_0)() &:= \langle \mathbf{a}, \langle \top \rangle \rangle \\ \mathcal{J}(g_1)() &:= \langle \mathbf{that}, \langle \top \rangle \rangle \\ \mathcal{J}(f_0)() &:= \langle \mathbf{dog}, \llbracket \{d_i : i \in \mathbb{N}\} \rrbracket \rangle \\ \mathcal{J}(f_1)() &:= \langle \mathbf{cat}, \llbracket \{c_i : i \in \mathbb{N}\} \rrbracket \rangle \\ \mathcal{J}(f_2)() &:= \langle \mathbf{mouse}, \llbracket \{m_i : i \in \mathbb{N}\} \rrbracket \rangle \\ \mathcal{J}(f_3)() &:= \langle \mathbf{cheese}, \llbracket \{h_i : i \in \mathbb{N}\} \rrbracket \rangle \\ \mathcal{J}(f_4)() &:= \langle \mathbf{saw}, \llbracket S \rrbracket \rangle \\ \mathcal{J}(f_5)() &:= \langle \mathbf{chased}, \llbracket C \rrbracket \rangle \\ \mathcal{J}(f_6)() &:= \langle \mathbf{ate}, \llbracket E \rrbracket \rangle \end{aligned}$$

There will be one mode of composition, which is binary. Let  $Y$  be the following linking aspect. For every unary concept it picks the unique minimal member and is defined on three binary concepts only, where  $Y(c)$  is that relation which contains  $V(c)$ , where  $V$  assigns the following critical sets to the concepts:

$$(4.105) \quad \begin{aligned} \llbracket E \rrbracket &\mapsto \{\langle m_0, h_0 \rangle\} \\ \llbracket C \rrbracket &\mapsto \{\langle c_0, m_0 \rangle\} \\ \llbracket S \rrbracket &\mapsto \{\langle d_0, c_0 \rangle\} \end{aligned}$$

(Recall  $V(c)$  is a set such that exactly one minimal member of  $c$  contains  $V(c)$ .  $Y(c)$  is defined to be that set.)

Now,  $\gamma(e, e')$  is defined if and only if either of the following holds:


- ①  $e = /a/$  and  $e'$  begins with  $/cheese/, /mouse/, /dog/,$  or  $/cat/$ .
- ②  $e \in \{/ate/, /saw/, /chased/\}$  and  $e'$  starts with  $/a_./$ .
- ③  $e = /that/$  and  $e'$  starts with  $/chased_./, /saw_./$  or  $/ate_./$ .

④  $e \in \{/cat/, /mouse/, /dog/, /cheese/\}$  and  $e'$  starts with  $/that_{\perp}/$ .

$$(4.106) \quad \mathcal{J}(m)(\langle e, m \rangle, \langle e', m' \rangle) := \begin{cases} \langle e \hat{\perp} e', m \otimes^Y m' \rangle & \text{if } \gamma(e, e') \\ \text{undefined} & \text{else} \end{cases}$$

So, the syntax is right regular. Without specifying too much detail let me note the first steps in the derivation.

$$(4.107) \quad \begin{aligned} & \langle \text{cheese}, \llbracket \{h_i : i \in \mathbb{N}\} \rrbracket \rangle \\ & \langle \text{a cheese}, \llbracket \{h_i : i \in \mathbb{N}\} \rrbracket \rangle \\ & \langle \text{ate a cheese}, \llbracket \{ \langle d_i, h_{i+1} \rangle : i \in \mathbb{N} \} \cup \{ \langle m_0, h_0 \rangle \} \rrbracket \rangle \\ & \langle \text{that ate a cheese}, \llbracket \{ \langle d_i, h_{i+1} \rangle : i \in \mathbb{N} \} \cup \{ \langle m_0, h_0 \rangle \} \rrbracket \rangle \\ & \langle \text{mouse that ate a cheese}, \llbracket \langle m_0, h_0 \rangle \rrbracket \rangle \\ & \langle \text{a mouse that ate a cheese}, \llbracket \langle m_0, h_0 \rangle \rrbracket \rangle \end{aligned}$$

At this point we get stuck; for we must now be able to combine two binary concepts. If we combine them the wrong way, instead of interpreting */a cat that chased a mouse that ate a cheese/* we interpret */a cat that chased a cheese that ate a mouse/*. As the embedding depth of relative clauses is unbounded there is no recipe for defining the linking aspect using critical sets as long as they do not exhaust the entire relation. So, we have to use a linking aspect instead. 

**Example 64.** We come to the first repair strategy. Leave everything as is with one exception. In the interpretation of  $m$ , quantify away the lower elements, always retaining a 1-concept.  $M$  is the domain of the model.

$$(4.108) \quad \mathcal{J}(m)(\langle e, m \rangle, \langle e', m' \rangle) := \begin{cases} \langle e \hat{\perp} e', \llbracket \mathbf{C}_1.(Y(m) \cap (M \times Y(m'))) \rrbracket \rangle & \text{if } \gamma(e, e') \text{ and } m \text{ is binary} \\ \langle e \hat{\perp} e', m \otimes^Y m' \rangle & \text{if } \gamma(e, e') \text{ and } m \text{ is unary} \\ \text{undefined} & \text{else} \end{cases}$$



The derivation now goes as follows.

$$\begin{aligned}
 & \langle \text{cheese}, \llbracket \{h_i : i \in \mathbb{N}\} \rrbracket \rangle \\
 & \langle \text{a cheese}, \llbracket \{h_i : i \in \mathbb{N}\} \rrbracket \rangle \\
 (4.109) \quad & \langle \text{ate a cheese}, \llbracket \{d_i : i \in \mathbb{N}\} \cup \{m_0\} \rrbracket \rangle \\
 & \langle \text{that ate a cheese}, \llbracket \{d_i : i \in \mathbb{N}\} \cup \{m_0\} \rrbracket \rangle \\
 & \langle \text{mouse that ate a cheese}, \llbracket \{m_0\} \rrbracket \rangle \\
 & \langle \text{a mouse that ate a cheese}, \llbracket \{m_0\} \rrbracket \rangle
 \end{aligned}$$

The step from the second to the third line is the crucial bit. We invoke the linking aspect on both concepts. The right hand side is unary, so we get the unique minimal member. The left hand side is the concept associated with one of the verbs, and by using the critical sets we align them such that the first column is subject and the second is object. We identify the object with the unary relation and quantify it away.

Thus, when we have processed one embedding we are back to a unary concept and can continue:

$$\begin{aligned}
 & \langle \text{chased a mouse that ate a cheese}, \llbracket \{c_0\} \rrbracket \rangle \\
 (4.110) \quad & \langle \text{that chased a mouse that ate a cheese}, \llbracket \{c_0\} \rrbracket \rangle \\
 & \langle \text{cat that chased a mouse that ate a cheese}, \llbracket \{c_0\} \rrbracket \rangle \\
 & \langle \text{a cat that chased a mouse that ate a cheese}, \llbracket \{c_0\} \rrbracket \rangle
 \end{aligned}$$

Thus, when we have processed one embedding we are back to a unary concept and can continue: The problem with this approach is that the intermediate objects are gone and cannot be referred to any more (say, with /the mouse that ate a cheese/).  $\odot$

**Example 65.** The second strategy uses profiling. Let  $P := \{t, b\}$ . The rule of combination is this. We assume that the subjects of verbs are assigned the profile  $t$ ; all other arguments are assigned  $b$ . When a verb is combined with an object, the object position is identified with the object with profile  $t$ , upon which the profile of this element is set to  $b$ . On the assumption that only one column has label  $t$ , we define the following linking algorithm. Let  $\langle t \cdot \vec{b}_1, R \rangle \in \mathcal{C}$  of length  $m$ ,  $\langle x \cdot \vec{b}_2, S \rangle \in \mathcal{D}$  of length  $n$ . Then we put

$$(4.111) \quad R \bar{\otimes} S := \{x \cdot \vec{y} \cdot \vec{z} : x \cdot \vec{y} \in R \text{ and } x \cdot \vec{z} \in S\}$$

This is almost like the Cartesian product, except that we take only the tuples that share the same first element, and eliminate its second occurrence. With respect to the profile, we proceed slightly differently. On the assumption that  $\langle t \cdot \vec{b}_1, R \rangle \in \mathcal{C}$  and  $\langle t \cdot \vec{b}_2, \mathcal{D} \rangle \in \mathcal{D}$  we put

$$(4.112) \quad \mathcal{C} \otimes^t \mathcal{D} := \llbracket \langle t \cdot \vec{b}_1 \cdot \vec{b}_2, R \bar{\otimes} S \rangle \rrbracket_{\mathcal{M}}$$

This is defined only if: (a) when both the concepts are at least unary, (b) when both profiles contain exactly one  $t$ . We extend this definition to the truth concept  $\mathcal{T}$  by putting

$$(4.113) \quad \mathcal{T} \otimes^t \mathcal{D} := \mathcal{D} \otimes^t \mathcal{T} := \mathcal{D}$$

All nouns denote concepts where the one minimal relation has profile  $t$ . And so we put

$$(4.114) \quad \mathcal{J}(m)(\langle e, m \rangle, \langle e', m' \rangle) := \begin{cases} \langle e \hat{\cap} e', m \otimes^t m' \rangle & \text{if } \gamma(e, e') \\ \text{undefined} & \text{else} \end{cases}$$

We denote the column with label  $t$  by underlining. The derivation begins as follows.

$$(4.115) \quad \begin{aligned} & \langle \text{cheese}, \llbracket \{ \underline{h}_i : i \in \mathbb{N} \} \rrbracket \rangle \\ & \langle \text{a cheese}, \llbracket \{ \underline{h}_i : i \in \mathbb{N} \} \rrbracket \rangle \\ & \langle \text{ate a cheese}, \llbracket \{ \langle \underline{d}_i, h_{i+1} \rangle : i \in \mathbb{N} \} \cup \{ \langle \underline{m}_0, h_0 \rangle \} \rrbracket \rangle \\ & \langle \text{that ate a cheese}, \llbracket \{ \langle \underline{d}_i, h_{i+1} \rangle : i \in \mathbb{N} \} \cup \{ \langle \underline{m}_0, h_0 \rangle \} \rrbracket \rangle \\ & \langle \text{mouse that ate a cheese}, \llbracket \langle \underline{m}_0, h_0 \rangle \rrbracket \rangle \\ & \langle \text{a mouse that ate a cheese}, \llbracket \langle \underline{m}_0, h_0 \rangle \rrbracket \rangle \end{aligned}$$

We have only one privileged member. We continue the derivation.

$$(4.116) \quad \begin{aligned} & \langle \text{chased a mouse that ate a cheese}, \llbracket \{ \langle \underline{c}_0, m_0, h_0 \rangle \} \rrbracket \rangle \\ & \langle \text{that chased a mouse that ate a cheese}, \llbracket \{ \langle \underline{c}_0, m_0, h_0 \rangle \} \rrbracket \rangle \\ & \langle \text{cat that chased a mouse that ate a cheese}, \\ & \quad \llbracket \{ \langle \underline{c}_0, m_0, h_0 \rangle \} \rrbracket \rangle \\ & \langle \text{a cat that chased a mouse that ate a cheese}, \\ & \quad \llbracket \{ \langle \underline{c}_0, m_0, h_0 \rangle \} \rrbracket \rangle \end{aligned}$$

The next step is to merge with /saw/:

(4.117) ⟨saw a cat that chased a mouse that ate a cheese,  
 $\llbracket \langle \underline{d}_0, c_0, m_0, h_0 \rangle \rrbracket \rangle$

And so on. Thus, the relations are growing in length but retain only one distinguished member.  $\odot$

The idea of profiling is not new. In formal semantics, referent systems (see [Vermeulen, 1995]) formalise a variant of profiling. Also Centering Theory implements a notion of profiling (see for example [Bittner, 2006] and references therein).

**Exercise 56.** Prove Proposition 4.21.



# Chapter 5

## Examples

THIS section will present some examples. The first example will be standard predicate logic. It will be shown that if semantics is based on concepts and not on relations then there must be a limit on the number of free variables. The second example will be a fragment (Montague size) of English. Finally, we shall indicate how the present approach allows to get insights into sentence structure.

### 5.1 Predicate Logic

This chapter is devoted to applications as well as examples. We begin by presenting standard predicate logic. In the section we shall give a grammar for predicate logic together with its standard interpretation(s), using sets of valuations or using relations. Then we shall turn to concept based interpretations. This will then be applied to natural language. Later in the chapter we shall show how the present assumptions on semantics (and syntax) allow to predict facts about sentential structure.

Recall from Section 4.2 the basic facts about predicate logic and its structures. In contrast to that section we do not deal with sorts; they do not add anything of significance. We start with a signature  $\langle \text{Rel}, \tau \rangle$ ,  $\text{Rel}$  a finite set of relation symbols and  $\tau : \text{Rel} \rightarrow \mathbb{N}$ . We shall as usual write  $\tau$  in place of  $\langle \text{Rel}, \tau \rangle$ . The alphabet is then the following set:  $A := \{ (, ), , 0, 1, x, \rightarrow, \neg, \forall, \wedge, \exists, \forall \} \cup \text{Rel}$ . We

assume that there are no function symbols. The arity of  $R \in \text{Rel}$  is given by  $\tau(R)$ . We shall first describe informally the formation rules of well-formed expressions and their meanings and then present a grammar of the interpreted language. The interpretation is based on a fixed 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  $\mathfrak{I}(R) \subseteq M^{\tau(R)}$ . A **valuation** is a function  $\beta : \{0, 1\}^* \rightarrow M$ . The set of all valuations is denoted by  $V$ .

In Section 4.2 we have provided meanings only for formulae. However, our alphabet is finite and we need an infinite array of variables. There is thus no other way than generating the set of variables from a finite base. This means, however, that we need to give some meaning to the variables. An **index** is a member of  $(0|1)^*$ , that is, string of  $/0/$  and  $/1/$ . The meaning of an index is the index itself. A **variable** is a sequence  $/x\vec{y}/$ , where  $\vec{y}$  is an index. The meaning of the variable is the function  $\vec{y}^* : \beta \mapsto \beta(\vec{y})$ . An **atomic formula** is an expression of the form  $/R^{\wedge}(\wedge \vec{v}_0, \wedge \vec{v}_1 \cdots \wedge \wedge \vec{v}_{\tau(R)-1})/$ , where the  $\vec{v}_i$  are variables. Its meaning is the set  $m(R) := \{\beta : \langle m(\vec{v}_0)(\beta), m(\vec{v}_1)(\beta), \dots, m(\vec{v}_{\tau(R)-1})(\beta) \rangle \in \mathfrak{I}(R)\}$ . Complex formulae are of the form  $/(\neg\varphi)/$ ,  $/(\varphi\wedge\chi)/$ ,  $/(\varphi\vee\chi)/$ ,  $/(\varphi\rightarrow\chi)/$ ,  $/(\exists x\vec{v})\varphi/$ ,  $/(\forall x\vec{v})\varphi/$ , where  $\vec{v}$  is an index,  $\varphi$  and  $\chi$  are formulae. The meaning of formulae has been spelled out earlier in Section 4.2. Thus the full language is  $L_\tau$ .

$$(5.1) \quad L_\tau := \begin{aligned} & \{ \langle \vec{v}, \vec{v} \rangle : \vec{v} \in (0|1)^* \} \\ & \cup \{ \langle x\vec{v}, \vec{v}^* \rangle : \vec{v} \in (0|1)^* \} \\ & \cup \{ \langle \varphi, [\varphi]_{\mathcal{M}} \rangle : \varphi \in \text{PL}_\tau \} \end{aligned}$$

(See (4.11) for a definition of  $[\cdot]_{\mathcal{M}}$ .) Now we shall present a grammar for that  $L_\tau$ . We shall use the following modes:

$$(5.2) \quad F := \{f_\emptyset, f_0, f_1, f_v, f_\neg, f_\wedge, f_\vee, f_\rightarrow, f_\exists, f_\forall\} \cup \{f_R : R \in \text{Rel}\}$$

The signature is  $\Omega : f_\emptyset \mapsto 0, f_1 \mapsto 1, f_2 \mapsto 1, f_v \mapsto 1, f_\neg \mapsto 1, f_\wedge \mapsto 2, f_\vee \mapsto 2, f_\rightarrow \mapsto 2, f_\exists \mapsto 2, f_\forall \mapsto 2, f_R \mapsto \tau(R)$ , where  $R \in \text{Rel}$ . First, we shall define the modes that build up variables. Recall that  $e^*(\beta) = \beta(e)$ , the function that is defined

on assignments and applies the assignment to the index.

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

The last rule seems dangerous since it seemingly converts any object  $m$  into a function  $m^*$  on assignments. However, the rules can only generate the pairs  $\langle \vec{v}, \vec{v} \rangle$ .

Next we turn to relations. Let  $R$  be a relation:

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

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

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

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  $v(m)$ .

$$(5.6) \quad \mathcal{C}(f_{\exists})(\langle e_0, m_0 \rangle, \langle e_1, m_1 \rangle) := \begin{cases} \langle (\wedge \exists e_0) \wedge e_1, \{\beta' : \text{exists } \beta \sim_{v(m_0)} \beta' : \beta \in m_1\} \rangle \\ \text{if } e_0 \text{ is a variable and } e_1 \text{ a formula} \\ \text{undefined else} \end{cases}$$


If  $M$  contains just one element then we put

$$(5.7) \quad \mathcal{C}(f_{\exists})(\langle e_0, m_0 \rangle, \langle e_1, m_1 \rangle) := \begin{cases} \langle (\wedge \exists e_0) \wedge e_1, m_1 \rangle \\ \text{if } e_0 \text{ is a variable and } e_1 \text{ a formula} \\ \text{undefined else} \end{cases}$$

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.

**Intermission 2.** One may have noticed that the grammar adds syncategorematic symbols other than brackets. In fact, all occurrences of logical and relation symbols are syncategorematic. This is unavoidable given the language  $L_{\tau}$ . For if  $/r/$  is a unary relation symbol  $/r(x)/$  is a formula, but the only part of it that is an expression is  $/x/$ , while  $/r/$  itself is not. This is a common dilemma. Montague has basically opted to make logical words in natural language syncategorematic. The



price is that it is thus unable to explain the meaning of /John walks and Pete talks./ in terms of the meaning of /and/ and the constituent sentences, but rather makes /and/ signal the application of a rule whose effect is to coordinate the sentences. 

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 language 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 arbitrary and not fixed in any way in advance (like it is in mathematical logic, for example); 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. (To my knowledge, the book [Monk, 1976] is a rare exception in actually using *variables* rather than metavariables.) If we want to give a semantics of predicate logic in terms of metavariables we must change the definitions rather substantially. Notice that the same issue arises in connection with programming languages. It used to be the case that variables had to have a specific format to make them distinct from other expressions. In many modern programming languages this is no longer required. Any expression that is not predefined can be used. Since the programmer is also free to define a host of other things, it turns out that it is highly context dependent whether or not a given sequence of letters actually denotes a variable.

There is certainly more than one way in which we can implement the semantics of predicate logic. Thus,  $L_\tau$  is one in many other formulations of predicate logic. Another way is described in Section 4.5. Let  $\mathcal{S} := \langle \mathcal{M}, \beta \rangle$  be a model. Based on the model  $\mathcal{S}$ , we perform a reduction of the formulae in the following way: write  $\varphi \equiv_{\mathcal{S}} \chi$  if

$$(5.8) \quad \langle \mathcal{M}, \beta \rangle \models \varphi \leftrightarrow \chi$$

This is an equivalence relation. Moreover, this is a congruence with respect to the standard boolean operations. This means that for  $\circ \in \{\vee, \wedge, \rightarrow\}$ :

$$(5.9) \quad \frac{\varphi \equiv_{\mathcal{S}} \chi \quad \varphi_1 \equiv_{\mathcal{S}} \chi_1 \quad \varphi_2 \equiv_{\mathcal{S}} \chi_2}{(\neg\varphi) \equiv_{\mathcal{S}} (\neg\chi) \quad (\varphi_1 \circ \varphi_2) \equiv_{\mathcal{S}} (\chi_1 \circ \chi_2)}$$

However, it is checked that the following does *not* hold.

$$(5.10) \quad \frac{\varphi \equiv_{\mathcal{S}} \chi}{(\exists x_i)\varphi \equiv_{\mathcal{S}} (\exists x_i)\chi}$$

Similarly, given just  $\mathcal{M}$ , write  $\varphi \equiv_{\mathcal{M}} \chi$  if for all  $\beta$

$$(5.11) \quad \langle \mathcal{M}, \beta \rangle \models \varphi \leftrightarrow \chi$$

This is equivalent to saying that for all  $\beta$ :

$$(5.12) \quad \langle \mathcal{M}, \beta \rangle \models \varphi \Leftrightarrow \langle \mathcal{M}, \beta \rangle \models \chi$$

This, in turn, is the same as  $[\varphi]_{\mathcal{M}} = [\chi]_{\mathcal{M}}$ . Finally, the denotation of a formula is not the set  $[\varphi]_{\mathcal{M}}$  but rather the set  $\{\chi : \varphi \equiv_{\mathcal{M}} \chi\}$ . This time not only the laws (5.9) hold (with  $\equiv_{\mathcal{M}}$  replacing  $\equiv_{\mathcal{S}}$ ) but we also have

$$(5.13) \quad \frac{\varphi \equiv_{\mathcal{M}} \chi}{(\exists x_i)\varphi \equiv_{\mathcal{M}} (\exists x_i)\chi}$$

I seize the opportunity to broaden the scope of the semantics somewhat. Let  $W$  be a set, the set of **worlds**. For every  $w \in W$  assume a model  $\mathcal{M}(w) = \langle \mathcal{M}(w), \mathfrak{I}(w), \beta(w) \rangle$ . This gives us an indexed family  $\mathcal{W} := \{\mathcal{M}(w) : w \in W\}$  of models. We write  $\varphi \equiv_{\mathcal{W}} \chi$  if for all  $w \in W$ :  $\varphi \equiv_{\mathcal{M}(w)} \chi$ . The laws (5.9) hold, but (5.10) need not hold.

The rationale behind this is that the family  $\mathcal{W}$  represents the space of all possibilities. We say that  $\varphi$  is **necessary** (in  $\mathcal{W}$ ) if  $\varphi \equiv_{\mathcal{W}} \top$ . (Here,  $\perp$  is any tautology, say,  $(\forall x)x=x$ ).  $\varphi$  is merely **possible** if  $\varphi \not\equiv_{\mathcal{W}} \perp$ . Let  $\Lambda$  be a first-order logic in the chosen signature  $\tau$ . Then for every formula  $\varphi \in L_{\tau}$  two choices arise: either it is inconsistent, that is, its negation is in  $\Lambda$ ; or it consistent, in which case there is a structure  $\mathcal{M}$  and a valuation  $\beta$  such that  $\langle \mathcal{M}, \beta \rangle \models \varphi$ . (See Chapter 4.2.) We can sharpen this somewhat. Say that a theory  $T$  is **maximally consistent** if  $T$  is consistent but there is no consistent  $U$  properly containing  $T$ . Let  $W$  be the set of maximally consistent sets and  $\langle \mathcal{M}(w), \beta(w) \rangle$  be a model such that for every  $\delta \in w$ :  $\langle \mathcal{M}(w), \beta(w) \rangle \models \delta$ . With this choice of  $\mathcal{W}$  we have that  $\varphi \equiv_{\mathcal{W}} \chi$  if and only if  $\varphi \leftrightarrow \chi$  is a theorem of predicate logic. In this model,  $\varphi$  is a necessary if it is logically true; and possible if logically consistent.

**Definition 5.1** A structure  $\mathcal{S} = \{\langle \mathcal{M}(w), \beta(w) \rangle : w \in W\}$  is **canonical** for a logic  $L$  if  $\varphi$  is necessary in  $\mathcal{S}$  if and only if  $\varphi$  is  $L$ -equivalent to  $\top$ , impossible in  $\mathcal{S}$  if and only if  $\varphi$  is  $L$ -equivalent to  $\perp$ , and possible otherwise.

This construction and result can be extended to other logics extending predicate logic. A particular case are *meaning postulates*.

**Example 66.** It is standardly assumed that /bachelor/ and /unmarried man/ are synonymous (ignoring presuppositions). There are two ways to implement this logically. One is to insert two unary predicate symbols, /r/ and /m/ and *define*

$$(5.14) \quad \mathbf{b}(x) := ((\neg r(x)) \wedge m(x))$$

This is basically a metalinguistic convention: it says that the string /b/ (which is not a relation symbol of our language), when followed by /( $x\vec{v}$ )/ is to be replaced by the sequence on the right, where /x/ is replaced by / $x\vec{v}$ /. Another way is to introduce three one place relation symbols, /b/, /m/ and /r/, and add the *meaning postulates*

$$(5.15) \quad (\forall x) (\mathbf{b}(x) \rightarrow ((\neg r(x)) \wedge m(x))) \quad (\forall x) (((\neg r(x)) \wedge m(x)) \rightarrow \mathbf{b}(x))$$

This means that our logic—call it  $L^+$ —is no longer predicate logic but a stronger logic. It is the least logic containing predicate logic and the two formulae of (5.15). The canonical structure for this logic consists in all models of the canonical structure for predicate logic in the new signature minus all the models where (5.15) does not hold.  $\odot$

Another point of extension is modal logics. Introduce a relation  $\triangleleft$  on the set  $W$ . Then pick  $w \in W$  and write

$$(5.16) \quad \langle \mathcal{W}, w \rangle \vDash \varphi \Leftrightarrow \langle \mathcal{M}(w), \beta(w) \rangle \vDash \varphi$$

Introduce a unary  $\Box$  operator on formulae and define

$$(5.17) \quad \langle \mathcal{W}, w \rangle \vDash (\Box\varphi) \text{ for all } u: \text{ if } w \triangleleft u \text{ then } \langle \mathcal{W}, u \rangle \vDash \varphi$$

This is the way in which Montague Semantics analyses propositional attitudes and tense, for example. We shall not have much to say on that topic, though. An alternative approach to intensionality is to add a new *sort*, that of a world, and make predicates relative to worlds.

**Exercise 57.** Spell out a grammar for the language  $\{\langle \varphi, (\varphi)_{\mathcal{M}} \rangle : \varphi \in L_{\tau}\}$ , adding interpretations for indices and variables as given in this section.

**Exercise 58.** Let  $L^+$  be the logic of Example 66. Let  $A$  be the set of formulae in (5.15). Say that a theory  $T$  is  $L^+$ -consistent if  $T \cup A$  is consistent. Use the Completeness Theorem to derive that there is a canonical structure  $\mathcal{S}$  for  $L^+$ .

**Exercise 59.** Define the following order on indices:

$$(5.18) \quad \varepsilon, \mathbf{0}, 1, \mathbf{00}, \mathbf{01}, \mathbf{10}, \mathbf{11}, \mathbf{000}, \dots$$

So,  $\vec{x}$  comes before  $\vec{y}$ , in symbols  $\vec{x} < \vec{y}$ , if and only if either  $\vec{x}$  is shorter than  $\vec{y}$  or  $\vec{x}$  and  $\vec{y}$  are of equal length and the binary number of  $\vec{x}$  is less than that of  $\vec{y}$ . Describe an algorithm to calculate from a number  $k$  the string  $\vec{x}$ , where  $\vec{x}$  has position  $k$  in the order  $<$ . Describe also the algorithm of the inverse to this mapping.

## 5.2 Concept Based Predicate Logic

In this section we shall explore the question how one can write a compositional grammar for predicate logic based on concepts. It will turn out that this is possible only if we restrict the language to a fragment based on finitely many variables. Whether or not the language is sorted is of no importance. Thus we ignore sorts and look at the following language:

$$(5.19) \quad \text{CL}_\tau := \{ \langle \varphi, \langle \varphi \rangle_{\mathcal{M}} \rangle : \varphi \in \text{PL}_\tau \}$$

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

$$(5.20) \quad f^\mu(\langle e_0, m_0 \rangle, \dots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1} \rangle) := \langle f_*^\varepsilon(\vec{e}), \langle f_*^\varepsilon(\vec{e}) \rangle_{\mathcal{M}} \rangle$$

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.

The principal result of this section is that for boundedly many variables this can be done, while for unboundedly many variables this is impossible in general. To start, let us 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  $\text{PL}_\tau^n$ , which is the fragment of predicate logic with relations in  $\tau$ , and variables from  $\{x_i : i < n\}$ . Functions will be omitted. Now fix a structure  $\mathcal{M} = \langle M, \mathfrak{F} \rangle$ . We put

$$(5.21) \quad \langle \varphi \rangle_{\mathcal{M}} := \llbracket (\varphi)_{\mathcal{M}} \rrbracket$$

We shall present an independent grammar for

$$(5.22) \quad \text{CL}_\tau^n = \{\langle \varphi, \langle \varphi \rangle_{\mathcal{M}} \rangle : \varphi \in \text{PL}_\tau^n\}$$

Define  $C := \{\langle \varphi \rangle_{\mathcal{M}} : \varphi \in \text{PL}_\tau^n\}$ , the expressive power of  $\text{CL}_\tau^n$ . It is clear that no relation of length  $> n$  can be minimal for any member of  $C$ . This is because there are only  $n$  different free variables to choose from, so they generate only  $n$ -ary relations. However,  $C$  not only contains concepts of length  $n$  but concepts of length  $k < n$  as well.

Let  $f : C \rightarrow \text{PL}_\tau^n$  be a function such that  $c = \langle f(c) \rangle_{\mathcal{M}}$ . Thus,  $f$  picks for each concept a formula defining it. For an arbitrary  $\chi \in \text{PL}_\tau^n$  the **type**  $\text{tp}(\chi)$  is a subset of  $\Pi_n$ , the set of permutations of  $n$  (see Appendix). It is defined by

$$(5.23) \quad \pi \in \text{tp}(\chi(\vec{x})) : \Leftrightarrow \mathcal{M} \models \chi(x_{\pi^{-1}(0)}, \dots, x_{\pi^{-1}(n-1)}) \leftrightarrow f(\langle \chi(\vec{x}) \rangle_{\mathcal{M}})$$

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). Given a type  $\pi$  and a concept  $c$  we define

$$(5.24) \quad f_\pi(c) := [x_{\pi(i)}/x_i : i < n]f(c)$$

Together with (5.23) this gives us for every  $\varphi \in \text{CL}_\tau^n$  and  $\pi \in \text{tp}(\varphi)$ :

$$(5.25) \quad \mathcal{M} \models \varphi \leftrightarrow f_\pi(\langle \varphi \rangle_{\mathcal{M}})$$

**Example 67.** Here is an example. Suppose we have a binary relation symbol  $r$  and we are looking at the language  $\text{PL}_\tau^2$ . The variables are called  $x_0$  (written here  $x_0$ ) and  $x_1$  (written here  $x_1$ ). Let  $c := \langle r(x_0, x_1) \rangle_{\mathcal{M}}$ . Then we also have

$$(5.26) \quad c = \langle r(x_1, x_0) \rangle_{\mathcal{M}}$$

Let  $f(c) = r(x_0, x_1)$ . Then the type of  $r(x_0, x_1)$  is the identity permutation, written  $()$ . However, the type of  $r(x_1, x_0)$  is the permutation  $\pi = (0\ 1)$ . For we have

$$(5.27) \quad f_\pi(c) = [x_1/x_0, x_0/x_1]r(x_0, x_1) = r(x_1, x_0)$$

And so we evidently have

$$(5.28) \quad \mathcal{M} \models r(x_1, x_0) \leftrightarrow f_\pi(\langle r(x_1, x_0) \rangle_{\mathcal{M}})$$

Similarly for more variables. A particular case to look at is where we have more variables than occur free in the formula, for example,  $PL_\tau^4$ . Here the type of  $r(x_0, x_1)$  consist both in  $()$  and in  $(2\ 3)$ , because the action on nonoccurring variables is irrelevant. Similarly, the types of  $r(x_1, x_0)$  are  $(0\ 1)$  and  $(0\ 1)(2\ 3)$ .  $\odot$

This finishes the preparations. We are ready to spell out the modes. They are given in Figure 5.1. In the definition, the followig functions are being used. For the existential quantifier we introduce the functions

$$(5.29) \quad \exists_\pi^i(c) := \langle (\exists x_i) f_\pi(c) \rangle_{\mathcal{M}}$$

For the universal quantifier we use

$$(5.30) \quad \forall_\pi^i(c) := \langle (\forall x_i) f_\pi(c) \rangle_{\mathcal{M}}$$

Now for the booleans.

$$(5.31) \quad \begin{aligned} N(c) &:= \langle (\neg f(c)) \rangle_{\mathcal{M}} \\ A_{\pi,\rho}(c, d) &:= \langle (f_\pi(c) \vee f_\rho(d)) \rangle_{\mathcal{M}} \\ C_{\pi,\rho}(c, d) &:= \langle (f_\pi(c) \wedge f_\rho(d)) \rangle_{\mathcal{M}} \\ I_{\pi,\rho}(c, d) &:= \langle (f_\pi(c) \rightarrow f_\rho(d)) \rangle_{\mathcal{M}} \end{aligned}$$

The modes are as follows: for every relation symbol  $R$  and every map  $\tau : n \rightarrow n$  (not necessarily injective) we pick a 0-ary mode  $f_\tau^R$ . For every  $i < n$  and every  $\pi \in \Pi_n$  we pick a unary mode  $f_{i,\pi}^\exists$  and a unary mode  $f_{i,\pi}^\forall$ . There will be 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  $\Omega_n$ . The interpretation  $\mathcal{J}_n$  is shown in Figure 5.1. Notice that  $i$  ranges over (not necessarily bijective or even injective) functions from  $n$  to  $n$ .

**Theorem 5.2** *The grammar  $G_n = \langle \Omega_n, \mathcal{J}_n \rangle$  is independent, context free and  $L(G_n) = CL_\tau^n$ .*

**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

Figure 5.1: The modes for  $CL_\tau^n$ 

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

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  $CL_\tau^n$ . This is done by induction. The inductive claim is that for every formula  $\varphi$  there is a term  $t$  such that  $\iota(t) = \langle \varphi, \llbracket \varphi \rrbracket_{\mathcal{M}} \rangle$ . The base case is

$$(5.33) \quad \varphi = R(\mathbf{x}_{j_0}, \cdots, \mathbf{x}_{j_{a(n)-1}})$$

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

$$(5.34) \quad \varphi = R(\mathbf{x}_{i(0)}, \cdots, \mathbf{x}_{i(a(n)-1)})$$

and so

$$(5.35) \quad \mathcal{J}(f_i^R) = \langle \varphi, \llbracket \varphi \rrbracket_{\mathcal{M}} \rangle$$

I perform only two of the inductive steps. Suppose for example that the formula has the form  $\langle e \vee e' \rangle$ . By inductive hypothesis there are analysis terms  $t$  and  $t'$  that unfold to  $\langle e, m \rangle$  and  $\langle e', m' \rangle$ , respectively. Let  $\pi$  be a type of  $e$  and  $\rho$  a type of  $e'$ . (Every formula has at least one type.) By inductive hypothesis,  $m = \langle e \rangle_{\mathcal{M}}$  and  $m' = \langle e' \rangle_{\mathcal{M}}$ . Then  $f_{\pi, \rho}^{\vee} tt'$  is defined and has exponent  $\langle e \vee e' \rangle$ . Then for the meaning we have by definition

$$(5.36) \quad \begin{aligned} & A_{\pi, \rho}(m, m') \\ &= \langle (f_{\pi}(m) \vee f_{\rho}(m')) \rangle_{\mathcal{M}} \\ &= \langle (e \vee e') \rangle_{\mathcal{M}} \end{aligned}$$

Next we deal with  $f_{i, \pi}^{\exists}$ . Suppose we have generated the sign  $\langle e, m \rangle$  using the term  $t$ . The induction hypothesis is that  $m = \langle e \rangle_{\mathcal{M}}$ . Assume that  $e$  has type  $\pi$ . Then from (5.25) we get

$$(5.37) \quad \mathcal{M} \models (\exists x_i) e \leftrightarrow (\exists x_i) f_{\pi}(\langle e \rangle_{\mathcal{M}})$$

and so

$$(5.38) \quad \langle (\exists x_i) e \rangle_{\mathcal{M}} = \langle (\exists x_i) f_{\pi}(\langle e \rangle_{\mathcal{M}}) \rangle_{\mathcal{M}} = \exists_{\pi}^i(m)$$

Then  $f_{i, \pi}^{\exists}$  can be applied to the sign and we get

$$(5.39) \quad \mathcal{J}_n(f_{i, \pi}^{\exists})(\langle e, m \rangle) = \langle (\exists x_i) e, \exists_{\pi}^i(m) \rangle$$

This completes the proof. □

The formulation of the semantics did not use linking aspects. They could in principle also be used, but it was easier to perform a definition by returning to the language  $CL_{\tau}^n$ . We were taking advantage of the fact that  $CL_{\tau}^n$  is unambiguous. In general, it is not possible to trade the linking aspect for functions to the exponents.

Let us discuss now the case where we have infinitely many variables. As I noted in Intermission 2, the language with infinitely many variables has the disadvantage that it must insert nontrivial syncategorematic symbols. Let us ignore that problem. Let us consider the language with  $\text{Rel} = \{\mathbf{r}\}$  and  $\tau(\mathbf{r}) = 2$ . The model is  $\mathcal{N} = \langle \mathbb{N}, \mathfrak{S} \rangle$ , with  $\mathfrak{S}(\mathbf{r}) = \{\langle i, i + 1 \rangle : i \in \mathbb{N}\}$ . We have three modes,  $f_{\emptyset}$  (zeroary),  $f_1$  and  $f_0$  (unary). Their interpretation is this (recall the definition of the verum



concept  $t$  as  $\llbracket\{\emptyset\}\rrbracket$ :

$$(5.40) \quad \mathfrak{I}(f_{\emptyset})() := \langle \mathbf{x}, t \rangle$$

$$(5.41) \quad \mathfrak{I}(f_0)(\langle e, m \rangle) := \begin{cases} \langle e^{\frown} \mathbf{0}, m \rangle & \text{if } e \text{ is a variable} \\ \text{undefined} & \text{else} \end{cases}$$

$$(5.42) \quad \mathfrak{I}(f_1)(\langle e, m \rangle) := \begin{cases} \langle e^{\frown} \mathbf{1}, m \rangle & \text{if } e \text{ is a variable} \\ \text{undefined} & \text{else} \end{cases}$$

Notice that we have this time generated variables from variables, to show that alternatives to introducing indices are possible. In fact, we are now generating the following language:

$$(5.43) \quad \text{CL}_{\tau} \cup \{ \langle \mathbf{x}\vec{u}, t : \vec{u} \in (\mathbf{0}\mathbf{1})^* \rangle \}$$

This language has two types of expressions: formulae and variables. The interpretation of variables is their range, and therefore the ‘truth’. Now we introduce the relation symbol by means of a binary mode:

$$(5.44) \quad f_r(\langle e, m \rangle, \langle e', m' \rangle) := \begin{cases} \langle \mathbf{r}(e^{\frown}, e'^{\frown}), \llbracket \langle i, i+1 \rangle : i \in \mathbb{N} \rrbracket_{\mathcal{N}} \rangle & \text{if } m \neq m' \text{ are variables} \\ \langle \mathbf{r}(e^{\frown}, e'^{\frown}), \llbracket \emptyset \rrbracket_{\mathcal{N}} \rangle & \text{if } m = m' \text{ are variables} \\ \text{undefined} & \text{else} \end{cases}$$

Define the following formulae.

$$(5.45) \quad \begin{aligned} \varphi_0 &:= \mathbf{r}(\mathbf{x}, \mathbf{x}\mathbf{0}) \\ \varphi_1 &:= (\mathbf{r}(\mathbf{x}, \mathbf{x}\mathbf{0}) \wedge \mathbf{r}(\mathbf{x}\mathbf{1}, \mathbf{x}\mathbf{0}\mathbf{0})) \\ \varphi_2 &:= ((\mathbf{r}(\mathbf{x}, \mathbf{x}\mathbf{0}) \wedge \mathbf{r}(\mathbf{x}\mathbf{1}, \mathbf{x}\mathbf{0}\mathbf{0})) \wedge (\mathbf{r}(\mathbf{x}\mathbf{0}\mathbf{0}, \mathbf{x}\mathbf{0}\mathbf{1}) \wedge \mathbf{r}(\mathbf{x}\mathbf{1}\mathbf{0}, \mathbf{x}\mathbf{1}\mathbf{1}))) \\ \varphi_3 &:= (((\mathbf{r}(\mathbf{x}, \mathbf{x}\mathbf{0}) \wedge \mathbf{r}(\mathbf{x}\mathbf{1}, \mathbf{x}\mathbf{0}\mathbf{0})) \wedge (\mathbf{r}(\mathbf{x}\mathbf{0}\mathbf{1}, \mathbf{x}\mathbf{1}\mathbf{0}) \\ &\quad \wedge \mathbf{r}(\mathbf{x}\mathbf{1}\mathbf{1}, \mathbf{x}\mathbf{0}\mathbf{0}\mathbf{0}))) \wedge ((\mathbf{r}(\mathbf{x}\mathbf{0}\mathbf{0}\mathbf{1}, \mathbf{x}\mathbf{0}\mathbf{1}\mathbf{0}) \\ &\quad \wedge \mathbf{r}(\mathbf{x}\mathbf{0}\mathbf{1}\mathbf{1}, \mathbf{x}\mathbf{1}\mathbf{0}\mathbf{0})) \wedge (\mathbf{r}(\mathbf{x}\mathbf{1}\mathbf{0}\mathbf{1}, \mathbf{x}\mathbf{1}\mathbf{1}\mathbf{0}) \wedge \mathbf{r}(\mathbf{x}\mathbf{1}\mathbf{1}\mathbf{1}, \mathbf{x}\mathbf{0}\mathbf{0}\mathbf{0}\mathbf{0})))) \end{aligned}$$

Also, define the following sets:

$$(5.46) \quad S_n := \{ \langle i, i+1, \dots, i+n-1 \rangle : i \in \mathbb{N} \}$$

For example,  $S_1$  is  $\mathbb{N}$ ,  $S_2$  consists of all pairs  $\langle 0, 1 \rangle$ ,  $\langle 1, 2 \rangle$ ,  $\langle 2, 3 \rangle$ , and so on, and  $S_3$  consists of the triples  $\langle 0, 1, 2 \rangle$ ,  $\langle 1, 2, 3 \rangle$ ,  $\langle 2, 3, 4 \rangle$ , and so on. The meaning of  $\varphi_0$

is  $\langle S_2 \rangle_{\mathcal{N}}$ , the meaning of  $\varphi_1$  is  $\llbracket S_2 \times S_2 \rrbracket_{\mathcal{N}}$ . The set of formulae we are interested in is a bit larger; it consists of all substitution instances of the  $\varphi_n$ . The following is easy to see.

**Lemma 5.3** *Let  $\chi$  be a substitution instance of  $\varphi_n$ . Either  $\chi$  is unsatisfiable in  $\mathcal{N}$  or  $\langle \chi \rangle_{\mathcal{N}}$  is the concept generated by a nontrivial product  $\prod_{k < p} S_{n(k)}$  for some numbers  $n(k) > 1$ .*

**Proof.** Clearly, some formulae are unsatisfiable, for example

$$(5.47) \quad ((r(x, x0) \wedge r(x0, x1)) \wedge (r(x, x1) \wedge r(x, x1))).$$

Now, let  $x < y$  if and only if  $\chi$  contains the clause  $r(x, y)$ . Say that  $x$  is of height 0 if there is no  $y$  such that  $y < x$ ; and of height  $n + 1$  if there is a  $y$  of height  $n$  such that  $y < x$ . Now we shall characterise all satisfying assignments. Suppose that  $x < y, y'$  and  $\beta$  a satisfying assignment; then  $\beta(y) = \beta(x) + 1$ , and  $\beta(y') = \beta(x) + 1$  from which  $\beta(y) = \beta(y')$ . Similarly, if  $x, x' < y$  then  $\beta(x) = \beta(x')$ . Let  $\approx_0$  be the identity. And let  $x \approx_{n+1} x'$  if for some  $y, y'$  such that  $y \approx_n y'$  either (a)  $y < x$  and  $y' < x'$  or (b)  $x < y$  and  $x' < y'$ ;  $x \approx x'$  is the union of all  $\approx_n$ . This is an equivalence relation. For  $\approx$ -equivalence classes  $A$  and  $B$  write  $A < B$  if there are  $x \in A$  and  $y \in B$  such that  $x < y$ . The relation  $<$  is linear on the classes. For assume  $A < B, B'$ . Then there are  $x, x' \in A$  and  $y \in B, y' \in B'$  such that  $x < y$  and  $x' < y'$ . Since  $x \approx x'$ , we have  $y \approx y'$ , by definition of  $\approx$ . Similarly we can show that if  $A, A' < B$  then  $A = A'$ . A valuation is now constructed as follows. For each class  $A$  which has no  $<$ -predecessor, pick a representative and assign to it any value. Then the values of the members of  $A$  must all be the same. Suppose that the values to members of  $A$  are known and are all identical to  $k$ ; let  $A < B$ . Then the value of every member of  $B$  is  $k + 1$ . By this recipe, the valuation is completely determined. Now let us turn to the concept defined by  $\chi$ . It is clear that when we pass to the concept all equivalence classes of  $\approx$  can be shrunk to one. All factors of the form  $S_1$  can be dropped. This gives the product representation.  $\square$

In particular, consider the following substitution instances.

$$\begin{aligned}
 \vartheta_0 &:= r(\mathbf{x}, \mathbf{x0}) \\
 \vartheta_1 &:= (r(\mathbf{x}, \mathbf{x0}) \wedge r(\mathbf{x0}, \mathbf{x1})) \\
 \vartheta_2 &:= ((r(\mathbf{x}, \mathbf{x0}) \wedge r(\mathbf{x0}, \mathbf{x1})) \wedge (r(\mathbf{x1}, \mathbf{x00}) \wedge r(\mathbf{x00}, \mathbf{x01}))) \\
 \vartheta_3 &:= (((r(\mathbf{x}, \mathbf{x0}) \wedge r(\mathbf{x0}, \mathbf{x1})) \wedge (r(\mathbf{x1}, \mathbf{x00}) \\
 &\quad \wedge r(\mathbf{x00}, \mathbf{x01}))) \wedge ((r(\mathbf{x01}, \mathbf{x10}) \\
 &\quad \wedge r(\mathbf{x10}, \mathbf{x11})) \wedge (r(\mathbf{x11}, \mathbf{x000}) \wedge r(\mathbf{x000}, \mathbf{x001}))))
 \end{aligned}
 \tag{5.48}$$

The meaning of these formulae is exactly  $\llbracket S_{2^{n+1}} \rrbracket_{\mathcal{N}}$ .

If  $\dot{g} = \langle g(0), \dots, g(k-1) \rangle$  is a vector of numbers, we put  $S_{\dot{g}} := \bigtimes_{i < k} S_{g(i)}$ . Let us look at the possible ways to assemble such formulae. We shall show that there is no way in which this sublanguage can be generated by a compositional context free interpreted grammar. This shall suffice for the following reason. The sublanguage is closed under taking subformulae; so if there is a grammar for the full language it must generate these formulae by means of other formulae of this kind. Hence if that is impossible, no grammar for the entire language exists.

Basically, for any context free grammar, the modes of composition must be to assemble some formulae and add some bounded material.

$$\begin{aligned}
 \mathcal{J}(f)(\langle e_0, m_0 \rangle, \dots, \langle e_{n-1}, m_{n-1} \rangle) \\
 := \langle \vec{x}_0 e_0 \vec{x}_1 \dots \vec{x}_{n-1} e_{n-1} \vec{x}_n, h_f(m_0, \dots, m_{n-1}) \rangle
 \end{aligned}
 \tag{5.49}$$

We may assume that  $m_i = \llbracket S_{\dot{g}(i)} \rrbracket_{\mathcal{N}}$  and that  $h_f(m_0, \dots, m_{n-1}) = \llbracket S_{\dot{g}(n)} \rrbracket_{\mathcal{N}}$ , where  $\dot{g}(0), \dots, \dot{g}(n)$  are vectors of natural numbers. In that way, the function  $h$  can be coded by the assignment

$$h_f^\star : \langle \dot{g}(0), \dots, \dot{g}(n-1) \rangle \mapsto \dot{g}(n)
 \tag{5.50}$$

Now the following can easily be verified.

**Lemma 5.4** *Suppose that  $n_i$  are numbers and that  $h_f^\star(n_0, \dots, n_{k-1})$  also is a number. Then  $h_f^\star(n_0, \dots, n_{k-1})$  can be any number between  $\max\{n_i : i < k\}$  and  $(\sum_{i < k} n_i) - (k - 1)$ .*

We now turn to an investigation of the morphology.

**Lemma 5.5** *Assume that  $\mathcal{J}(f)$  is as in (5.49). Then for given  $e'$  there is at most one vector  $\vec{e} = \langle e_i : i < \Omega(f) \rangle$  such that  $f^\varepsilon(\vec{e}) = e'$ .*

**Proof.** Let  $n = \Omega(f)$ . Assume that  $f^\varepsilon(\vec{e}) = f^\varepsilon(\vec{c})$  for some vector  $\vec{c} = \langle c_i : i < n \rangle$ . Then we have

$$(5.51) \quad \vec{x}_0 e_0 \vec{x}_1 e_1 \vec{x}_2 \cdots \vec{x}_{n-1} e_{n-1} \vec{x}_n = \vec{x}_0 c_0 \vec{x}_1 c_1 \vec{x}_2 \cdots \vec{x}_{n-1} c_{n-1} \vec{x}_n$$

From this it follows that

$$(5.52) \quad e_0 \vec{x}_1 e_1 \vec{x}_2 \cdots \vec{x}_{n-1} e_{n-1} \vec{x}_n = c_0 \vec{x}_1 c_1 \vec{x}_2 \cdots \vec{x}_{n-1} c_{n-1} \vec{x}_n$$

Suppose first that  $e_0$  and  $c_0$  are formulae. It is a property of this language that no prefix of a formula is a formula. Hence  $e_0 = c_0$  and so

$$(5.53) \quad \vec{x}_1 e_1 \vec{x}_2 \cdots \vec{x}_{n-1} e_{n-1} \vec{x}_n = \vec{x}_1 c_1 \vec{x}_2 \cdots \vec{x}_{n-1} c_{n-1} \vec{x}_n$$

Now assume that  $e_0$  is not a formula. Then it is a variable and so of the form  $\mathbf{x}\vec{u}$ , where  $\vec{u}$  is a binary string. In this case, since also  $e_1$  is either a variable or a formula,  $\vec{x}_1$  must contain a prefix that finishes the occurrence of the variable that  $e_0$  begins. It does the same with  $c_0$ ; thus,  $e_0 = c_0$ . Repeat this argument  $n - 1$  times.  $\square$

Finally, let  $\langle \vec{u}, \vec{v} \rangle$  be an occurrence of  $\vec{x}$  in  $\vec{z} = \vec{u}\vec{x}\vec{v}$ . The **embedding depth** of this occurrence of  $\vec{x}$  is defined to be the number of occurrences of and opening bracket minus the number of occurrences of a closing bracket in  $\vec{u}$ . Notice that in  $\varphi_n$  every atomic subformula has embedding depth  $n$ .

**Lemma 5.6** *Let  $\chi$  be a formula with an occurrence of depth  $d$  in  $\varphi_n$ . Then  $\chi$  is a substitution instance of  $\varphi_{n-d}$ .*

**Proof.** By induction on  $n - d$ . Let  $n = d$ . Since no formula has embedding depth  $> n$ , the formula is atomic and so a substitution instance of  $\varphi_0$ . Now let the claim be shown for  $n - d$ . We show it for  $n - d + 1$ . Let us be given an occurrence  $\langle \vec{u}, \vec{v} \rangle$  of  $\chi$ . Then  $\chi$  begins with an opening bracket (since no atomic formula has embedding depth  $n - d + 1$ ). Thus, it is easily seen that  $\chi = (\vec{z}_0 \wedge \vec{z}_1)$ , where  $\vec{z}_0$  and  $\vec{z}_1$  are subformulae of embedding depth  $n - d$ . By inductive hypothesis, they are substitution instances of  $\varphi_{n-d}$ . Then  $\chi$  is a substitution instance of  $\varphi_{n-d+1}$ .  $\square$

Thus, with  $\mathcal{J}$  defined as in (5.49) let  $\mu_f$  be the largest of the bracket balances of  $\vec{x}_0\vec{x}_1 \cdots \vec{x}_i$ ,  $i < n$ . Now, if  $f^e(e_0, \dots, e_{n-1}) = \varphi_n$ , we conclude that the embedding depth of the occurrences of the  $e_i$  in  $\varphi_n$  are less than or equal to  $n - \mu_f$ . By choosing  $n$  large enough we can make the  $e_i$  to be of any minimal length we want.

Let now  $G$  be any context free compositional interpreted grammar for the language. Define

$$(5.54) \quad \mu_G := \max\{\mu_f : f \in F\} \quad \alpha_G := \max\{\Omega(f) : f \in F\}$$

Make  $n$  large enough so that  $n^* := 2^{n-\mu_G} + 1 > \alpha_G + \text{card } F$ . For every  $f \in F$ , let  $\dot{\nu}_f := h_f^*(n^*, \dots, n^*)$ . By choice of  $n^*$  there is a number  $j^*$  between  $n^*$  and  $2n^* - 1$  which is not of the form  $\dot{\nu}_f$ . (If  $\dot{\nu}_f$  is not a number, that is anyhow the case.) Next let  $\psi_k$  be the substitution into  $\vartheta_{n-\mu_G}$  such that the names of the variables are shifted by  $k$  in the order  $<$  (see Exercise 59). Since this shift is injective, the meaning of  $\psi_k$  is the same as that of  $\vartheta_{n-\mu_G}$ , which is  $\llbracket S_{2n^*} \rrbracket$ . Now we define the following sequence of formulae:

$$(5.55) \quad \begin{aligned} \chi^\circ(0) &:= \psi_0 & \chi^\bullet(0) &:= \psi_{j^*-n^*} \\ \chi^\circ(n+1) &:= (\bigwedge \chi^\circ(n) \wedge \chi^\circ(n) \wedge) & \chi^\bullet(n+1) &:= (\bigwedge \chi^\bullet(n) \wedge \chi^\bullet(n) \wedge) \end{aligned}$$

Finally, let  $\zeta := (\chi^\circ(\mu_G) \wedge \chi^\bullet(\mu_G))$ . Its meaning is  $\llbracket S_{j^*} \rrbracket$ . (For  $\chi^\circ(\mu_G)$  contains the first  $n^*$  variables, and  $\chi^\bullet(\mu_G)$  contains this set shifted by  $j^* - n^*$  (which is a number  $< n^*$ ). Their conjunction therefore contains the first  $j^*$  variables.

We show that  $\langle \zeta, \langle \zeta \rangle_{\mathcal{N}} \rangle$  cannot be generated in  $G$ . For assume that it is the value of the term  $ft_0 \cdots t_{n-1}$ . Then  $\zeta$  has a decomposition as follows.

$$(5.56) \quad \zeta = \vec{x}_0 e_0 \vec{x}_1 e_1 \vec{x}_2 \cdots \vec{x}_{n-1} e_{n-1} \vec{x}$$

As we have seen, the  $e_i$  must be subformulae. Now, we may assume that  $e_i \neq \zeta$  (or else  $\zeta = e_0$ , and then we must obviously find a way to generate  $e_0$  using another function). And so the  $e_i$  are subformula of either  $\chi^\circ(\mu_G)$  or of  $\chi^\bullet(\mu_G)$ . As they are of embedding depth at most  $\mu_G$  they have the form  $\chi^\circ(d)$  or  $\chi^\bullet(d)$  for some  $d$ . Hence their meaning is  $\llbracket S_{n^*} \rrbracket$ . The denotation of the term  $ft_0 \cdots t_{n-1}$  is of the form  $\llbracket S_k \rrbracket$  where  $k = h_f^*(n^*, \dots, n^*)$ . However,  $\zeta$  has the meaning  $\llbracket S_{j^*} \rrbracket$ , which is not of this form. This completes the proof.

**Theorem 5.7** *There are models and signatures for which  $\text{CL}_\tau$  has no compositional interpreted context free grammar.*

It is perhaps worthwhile saying something about the significance of this result. In generative grammar it has been observed that there are constituents that serve as a bottleneck in syntax, called *phases*. In the earlier fragment of [Chomsky, 1986], the CP- and DP-constituents had the property that, unlike VPs, they could not be adjoined to arbitrarily. While the existence of phases has always been a mystery, here we find an indication as to why such bottlenecks must exist. Since meanings are not of kind we saw in the previous section, but rather have the combinatorics of concepts, there is a limit on how many elements we can have in storage. One way of calibrating the idea of storage is to calculate the number of free variables occurring in a formula.

**Exercise 60.** The function for concept negation did not depend on the type of the formula, while the disjunction, conjunction and implication depended on the types of both arguments. A closer analysis reveals that for an  $n$ -ary boolean operator the concept function depends on all  $n$  types; it is however enough to assume functions that depend only on  $n - 1$  arguments. Can you give a general solution how to lift an  $n$ -ary operator to concepts using  $n - 1$  type parameters rather than  $n$ ? Perform this reduction for  $A_{\pi;\rho}$ ,  $C_{\pi;\rho}$  and  $I_{\pi;\rho}$ . Can you see why negation is independent of its unique argument?

**Exercise 61.** What happens if we allow functions in the primitive vocabulary of predicate logic?

**Exercise 62.** Modify the above proof of Theorem 5.7 to the case where the language is as follows (cf. the definition of  $L_\tau$  of the previous section):

$$(5.57) \quad \text{CL}_\tau \cup \{ \langle \vec{v}, \vec{v} \rangle : \vec{v} \in (\mathbf{0|1})^* \}$$

**Exercise 63.** Show that Theorem 5.7 would also hold if we allowed to introduce an arbitrary finite set of categories. (Assuming, of course, that the grammar is independent in all three components.)

**Exercise 64.** Here is a variation on the formulae defined above. Define  $\eta_n$  as follows.

$$\begin{aligned}
 \eta_0 &:= r(x, x0) \\
 \eta_1 &:= (r(x, x0) \wedge r(x1, x00)) \\
 \eta_2 &:= ((r(x, x0) \wedge r(x1, x00)) \wedge r(x01, x10)) \\
 \eta_3 &:= (((r(x, x0) \wedge r(x1, x00)) \wedge r(x01, x10)) \wedge r(x11, 000))
 \end{aligned}
 \tag{5.58}$$

Show that no compositional context free interpreted grammar exists that generates all the pairs  $\langle s(\eta_n), \langle s(\eta_n) \rangle \rangle$ , where  $s$  is a substitution (together with all pairs  $\langle \mathbf{x}\vec{v}, \llbracket \{\emptyset\} \rrbracket_{\mathcal{N}} \rangle$ ).

## 5.3 A Fragment of English

In this section we shall show by way of examples in which way one can overcome the limitations of concepts. 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 standard, relation based, 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 time 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 uniquely identifiable through its sort. Only  $x$  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 in our standard model  $\mathcal{M}$

$$(5.59) \quad \mathcal{M} \models \text{hit}'(t, w, x, y) \rightarrow \alpha(t, w, x), \quad \mathcal{M} \not\models \text{hit}'(t, w, x, y) \rightarrow \alpha(t, w, y)$$

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

$$(5.60) \quad \mathcal{M} \models \Box(\text{hit}'(x, y) \rightarrow \alpha(x)), \quad \mathcal{M} \not\models \Box(\text{hit}'(x, y) \rightarrow \alpha(y))$$

But these differences are superficial. Let us suppose that something like (5.59) holds. However, as the model we are using is characteristic (all that is logically true is true in it, all that is logically false is false in it), 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  $\pi([\varphi]_{\mathcal{M}})$ ):

$$(5.61) \quad \begin{aligned} [\text{hit}'(t, w, x, y)]_{\mathcal{M}} &\subseteq [\alpha(t, w, x)]_{\mathcal{M}} \times M_e \\ \pi_{(23)}[\text{hit}'(t, w, x, y)]_{\mathcal{M}} &\not\subseteq [\alpha(t, w, x)]_{\mathcal{M}} \times M_e \end{aligned}$$

The formula  $\alpha(t, w, x)$  does not suffer from the same combinatorial ambiguity. Thus, the concept  $\langle\alpha(t, w, x)\rangle_{\mathcal{M}}$  has only *one* minimal member in its type. 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 68.** There are three sorts,  $e$ ,  $w$  and  $t$ . Assume that  $M_e = \{a, b, c\}$ ,  $M_w = \{w_0, w_1\}$ , and  $M_t = \{t_0, t_1\}$ .

$$(5.62) \quad \begin{aligned} [\alpha(t, w, x)]_{\mathcal{M}} = &\{\langle t_0, w_0, a \rangle, \langle t_0, w_0, b \rangle, \langle t_0, w_0, c \rangle, \langle t_0, w_1, a \rangle, \\ &\langle t_1, w_0, b \rangle, \langle t_1, w_0, c \rangle\} \end{aligned}$$

$$(5.63) \quad \begin{aligned} [\text{hit}'(t, w, x, y)]_{\mathcal{M}} = &\{\langle t_0, w_0, a, a \rangle, \langle t_0, w_0, a, b \rangle, \langle t_0, w_0, b, a \rangle, \\ &\langle t_0, w_0, a, c \rangle, \langle t_1, w_0, c, a \rangle, \langle t_1, w_0, c, b \rangle\} \end{aligned}$$

In this model (5.61) is satisfied. This means that we can discriminate the two minimal members  $T_0$  and  $T_1$  of the concept:

$$(5.64) \quad \begin{aligned} T_0 := &\{\langle t_0, w_0, a, a \rangle, \langle t_0, w_0, a, b \rangle, \langle t_0, w_0, b, a \rangle, \\ &\langle t_0, w_0, a, c \rangle, \langle t_1, w_0, c, a \rangle, \langle t_1, w_0, c, b \rangle\}, \\ T_1 := &\{\langle t_0, w_0, a, a \rangle, \langle t_0, w_0, b, a \rangle, \langle t_0, w_0, a, b \rangle, \\ &\langle t_0, w_0, c, a \rangle, \langle t_1, w_0, a, c \rangle, \langle t_1, w_0, b, c \rangle\} \end{aligned}$$



Indeed,  $T_1$  contains  $\langle t_1, w_0, a, b \rangle$ , and this is not contained in the set  $[\alpha(t, w, x)]_{\mathcal{M}} \times M_e$ .

$$(5.65) \quad [\alpha(t, w, x)]_{\mathcal{M}} \times M_e = \{ \langle t_0, w_0, a, a \rangle, \langle t_0, w_0, a, b \rangle, \langle t_0, w_0, a, c \rangle, \\ \langle t_0, w_0, b, a \rangle, \langle t_0, w_0, b, b \rangle, \langle t_0, w_0, b, c \rangle, \langle t_0, w_0, c, a \rangle, \\ \langle t_0, w_0, c, b \rangle, \langle t_0, w_0, c, c \rangle, \langle t_0, w_1, a, a \rangle, \langle t_0, w_1, a, b \rangle, \\ \langle t_0, w_1, a, c \rangle, \langle t_1, w_0, b, a \rangle, \langle t_1, w_0, b, b \rangle, \langle t_1, w_0, b, c \rangle, \\ \langle t_1, w_0, c, a \rangle, \langle t_1, w_0, c, b \rangle, \langle t_1, w_0, c, c \rangle \}$$

Notice how the intensionality does real work. For in  $w_0$  at  $t_0$  every object has property  $\alpha$ . If we had to define our minimal member only here, there would be no way to distinguish the arguments. For example, suppose that at  $w_0$  and  $t_0$ , everybody is such that he or she is moving and exerting some force. Still it should not follow that everybody is hitting someone. They could, for example, push a car uphill. Thus, we need to make reference to other worlds. Additionally, of course, in the entire space of worlds there must be one where the concepts really *is* nonsymmetrical, otherwise (5.61) could not be used to discriminate the arguments.  $\odot$

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  $d_{ij}^k$  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. Let  $\sigma = \langle e, c, m \rangle$  and  $e' = \langle e', c', m' \rangle$ .

$$(5.66) \quad \begin{aligned} \mathcal{D}(f_0)() &:= \langle \text{John, NP, } \{a\} \rangle \\ \mathcal{D}(f_1)() &:= \langle \text{Paul, NP, } \{b\} \rangle \\ \mathcal{D}(f_2)() &:= \langle \text{hits, V, } \langle \text{hit}'(t, w, x, y) \rangle_{\mathcal{M}} \rangle \\ \mathcal{D}(f_3)(\sigma, \sigma') &:= \begin{cases} \langle e \hat{\ } \square \hat{\ } e', \text{VP, } \langle \text{C}_2 \cdot \text{C}_4 \cdot (Y(m) \times Y(m') \cap d_{24}^5) \rangle_{\mathcal{M}} \rangle \\ \text{if } c = \text{V and } c' = \text{NP} \\ \text{undefined else} \end{cases} \\ \mathcal{D}(f_4)(\sigma, \sigma') &:= \begin{cases} \langle e' \hat{\ } \square \hat{\ } e \hat{\ } ., \text{S, } \langle \text{C}_0 \cdot \text{C}_1 \cdot \text{C}_2 \cdot \text{C}_3 \cdot (Y(m) \times Y(m') \cap d_{23}^4) \rangle_{\mathcal{M}} \rangle \\ \text{if } c = \text{VP and } c' = \text{NP} \\ \text{undefined else} \end{cases} \end{aligned}$$

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. Let us see how that works. The 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.

$$\begin{aligned}
 \iota_G(f_3 f_2 f_1) &= \mathcal{D}(f_3)(\langle \text{hits}, \text{V}, R \rangle, \langle \text{Paul}, \text{NP}, \{b\} \rangle) \\
 (5.67) \quad &= \langle \text{hits} \square \square \text{Paul}, \text{VP}, \ll \mathbf{C}_2 \cdot \mathbf{C}_4 \cdot (Y(m) \times Y(m') \cap d_{24}^5) \gg_{\mathcal{M}} \rangle \\
 &= \langle \text{hits Paul}, \text{VP}, \ll \{ \langle t_0, w_0, a \rangle, \langle t_1, w_0, c \rangle \} \gg_{\mathcal{M}} \rangle
 \end{aligned}$$

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

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

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

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

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  $x_2 = x_4$ .

$$(5.70) \quad Y(m) \times Y(m') \cap d_{24}^5 = \{ \langle t_0, w_0, b, a, b \rangle, \langle t_1, w_0, b, c, b \rangle \}$$

Finally, we remove the columns 2 and 4:

$$(5.71) \quad \mathbf{C}_2 \cdot \mathbf{C}_4 \cdot Y(m) \times Y(m') \cap d_{24}^5 = \{ \langle t_0, w_0, a \rangle, \langle t_1, w_0, c \rangle \}$$

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 (with  $Y(m)$  defined below):

$$\begin{aligned}
 \iota_G(f_4 f_3 f_2 f_1 f_0) &= \mathcal{D}(f_4)(\iota_G(f_3 f_2 f_1), \iota_G(f_0)) \\
 &= \mathcal{D}(f_4)(\langle \text{hits Paul}, \text{VP}, \ll \{ \langle t_0, w_0, a \rangle, \langle t_1, w_0, c \rangle \} \gg_{\mathcal{M}} \rangle, \\
 &\quad \langle \text{John}, \text{NP}, \{a\} \rangle) \\
 (5.72) \quad &= \langle \text{John} \square \square \text{hits Paul} \cdot, \text{S}, \\
 &\quad \ll \mathbf{C}_0 \cdot \mathbf{C}_1 \cdot \mathbf{C}_2 \cdot \mathbf{C}_3 \cdot (Y(m) \times Y(\ll \{a\} \gg_{\mathcal{M}}) \cap d_{23}^4) \gg_{\mathcal{M}} \rangle \\
 &= \langle \text{John hits Paul} \cdot, \text{S}, \{\emptyset\} \rangle
 \end{aligned}$$

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  $<$  worlds  $<$  entities. (This does not follow, by the way, from our assumption on how it orders the minimal members of the concept of hitting!) Then

$$(5.73) \quad Y(m) = \{\langle t_0, w_0, a \rangle, \langle t_1, w_0, c \rangle\}$$

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

$$(5.74) \quad Y(m) \times Y(\langle\langle a \rangle\rangle_{\mathcal{M}}) = \{\langle t_0, w_0, a, a \rangle, \langle t_1, w_0, c, a \rangle\}$$

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

$$(5.75) \quad Y(m) \times Y(\langle\langle a \rangle\rangle_{\mathcal{M}}) \cap d_{23}^4 = \{\langle t_0, w_0, a, a \rangle\}$$

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

$$(5.76) \quad C_0.C_1.C_2.C_3.Y(m) \times Y(\langle\langle a \rangle\rangle_{\mathcal{M}}) \cap d_{23}^4 = \{\langle \rangle\}$$

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,

$$(5.77) \quad \begin{aligned} & [\varphi(t, w, x, y, z)]_{\mathcal{M}} \subseteq [\alpha(t, w, x)]_{\mathcal{M}} \times M_e \times M_e \\ \pi_{(23)}[\varphi(t, w, x, y, z)]_{\mathcal{M}} & \not\subseteq [\alpha(t, w, y)]_{\mathcal{M}} \times M_e \times M_e \\ \pi_{(24)}[\varphi(t, w, x, y, z)]_{\mathcal{M}} & \not\subseteq [\alpha(t, w, z)]_{\mathcal{M}} \times M_e \times M_e \end{aligned}$$

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 5.8** 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 trivial. Here, a permutation  $\pi$  is sortally trivial 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

$$(5.78) \quad \mathbb{S}(c) := \{\vec{x}_\pi : \pi \in \Xi\}$$

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 69.** 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  $[\text{see}'(y, x, t, w)]_{\mathcal{M}}$ .  $\otimes$

**Example 70.** Assume one sort  $e$ , and  $M_e = \{a, b, c\}$ . Let

$$(5.79) \quad R = \{\langle a, b, c \rangle, \langle a, c, b \rangle, \langle b, a, b \rangle, \langle b, b, a \rangle\}$$

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  $\{\langle a, b, c \rangle\}$ .  $\otimes$

## 5.4 Concepts and LF

It seems that the introduction of concepts actually made matters worse. To get meanings in a compositional way is not at all straightforward. 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 what the algorithm of [Kamp and Reyle, 1993] factually does. It translates the sentence (5.80) not directly, but via surface indexing.

(5.80) 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 (5.81) rather than (5.80). 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. Making this absolutely clear is essential.

(5.81) A<sub>1</sub> 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 /man<sub>0</sub>/ has a different translation than /man<sub>1</sub>/. Therefore, in order for the proposed algorithm to work, we must assume that the grammar generates entries of the following form:

(5.82) ⟨man<sub>0</sub>, man'(x<sub>0</sub>)⟩, ⟨man<sub>1</sub>, man'(x<sub>1</sub>)⟩, ⟨man<sub>2</sub>, man'(x<sub>2</sub>)⟩, ⋯

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

(5.83) ⟨man<sub>0</sub>, man'(x<sub>0</sub>)⟩

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

(5.84) 
$$\begin{array}{ll} \langle \text{sees}_{(0,0)}, \text{see}'(x_0, x_0) \rangle, & \langle \text{sees}_{(1,0)}, \text{see}'(x_1, x_0) \rangle, \\ & \langle \text{sees}_{(2,0)}, \text{see}'(x_2, x_0) \rangle, \dots \\ \langle \text{sees}_{(0,1)}, \text{see}'(x_0, x_1) \rangle, & \langle \text{sees}_{(1,1)}, \text{see}'(x_1, x_1) \rangle, \\ & \langle \text{sees}_{(2,1)}, \text{see}'(x_2, x_1) \rangle, \dots \\ \langle \text{sees}_{(0,2)}, \text{see}'(x_0, x_2) \rangle, & \langle \text{sees}_{(1,2)}, \text{see}'(x_1, x_2) \rangle, \\ & \langle \text{sees}_{(2,2)}, \text{see}'(x_2, x_2) \rangle, \dots \\ \dots & \dots \end{array}$$

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/. 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. Thus we rather have the following signs

$$(5.85) \quad \langle \text{man}, \text{man}'(x_0) \rangle, \langle \text{man}, \text{man}'(x_1) \rangle, \langle \text{man}, \text{man}'(x_2) \rangle, \dots$$

$$(5.86) \quad \begin{array}{lll} \langle \text{sees}, \text{see}'(x_0, x_0) \rangle, & \langle \text{sees}, \text{see}'(x_1, x_0) \rangle, & \langle \text{sees}, \text{see}'(x_2, x_0) \rangle, \dots \\ \langle \text{sees}, \text{see}'(x_0, x_1) \rangle, & \langle \text{sees}, \text{see}'(x_1, x_1) \rangle, & \langle \text{sees}, \text{see}'(x_2, x_1) \rangle, \dots \\ \langle \text{sees}, \text{see}'(x_0, x_2) \rangle, & \langle \text{sees}, \text{see}'(x_1, x_2) \rangle, & \langle \text{sees}, \text{see}'(x_2, x_2) \rangle, \dots \\ \dots & \dots & \dots \end{array}$$

This means that the name of the actual variable has become immaterial. This is essentially what is meant by the Principle of Alphabetical Innocence.<sup>1</sup>

**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  $\varphi$ .*

It is possible to derive this from our postulates on meaning. However, it is worth stating on its own because it allows us to decide in a simple way whether a semantics is properly desyntactified. We shall apply the principle to the case at hand. It means that none of the predicate logical formulae properly capture the meaning of /man/ or /see/. For if the meaning of /man/ was expressed by, say,  $\text{man}'(x_0)$ , then we should have

$$(5.87) \quad \text{man}'(x_0) \leftrightarrow \text{man}'(x_1)$$

<sup>1</sup>This name is due to Kit Fine, which he used during a lecture at UCLA.

But this is false in the standard semantics for predicate logic. Notice that even a formula such as  $\bigvee_{i \in \mathbb{N}} \text{man}'(x_i)$  is no good, since it is not invariant under shift:  $s : x_i \mapsto x_{i+1}$ .

$$(5.88) \quad \neq \bigvee_{i \in \mathbb{N}} \text{man}'(x_i) \leftrightarrow s \left( \bigvee_{i \in \mathbb{N}} \text{man}'(x_i) \right) = \bigvee_{i \in \mathbb{N} - \{0\}} \text{man}'(x_i)$$

We can now see why an approach of the sort advocated in generative grammar is no solution. Take, for example, the semantics of [Heim and Kratzer, 1998]. For the purposes of presentation, I take a very simple example. The analysis of the sentence /every man runs/ proceeds as follows. The LF associated with this sentence is

$$(5.89) \quad \text{every man } [8 \ [t_8 \ \text{runs}]]$$

This is interpreted bottom up. Notice that  $\text{man}'$  is the same as  $\lambda x_0.\text{man}'(x_0)$ , and  $\text{run}'$  the same as  $\lambda x_0.\text{run}'(x_0)$ :

$$(5.90) \quad \frac{\frac{\lambda P.\lambda Q.\forall x_0.P(x_0) \rightarrow Q(x_0) \quad \text{man}' \quad \lambda P.\lambda x_8.P \quad x_8 \quad \text{run}'}{\lambda Q.\text{man}'(x_0) \rightarrow Q(x_0)} \quad \frac{\vdots \quad \text{run}'(x_8)}{\lambda x_8.\text{run}'(x_8)}}{\forall x_0.\text{man}'(x_0) \rightarrow \text{run}'(x_0)}$$

Essentially, the semantics does two things in sequence: first, the functions are applied to some variables, in this case  $x_8$ . The net effect of this is that the variable is *displayed*. In generative grammar this is done because variables are the interpretation of traces. This is the step of VP formation. The VP then has as its interpretation an open formula. Next, a step of function *abstraction* is performed. The element denoted by '8' does nothing but to abstract the variable  $x_8$ . Finally, the quantifier, being a function, takes the abstracted form as its argument.

The success of this proposal lies in the possibility to display and (re)abstract variables at each step of the derivation. This however demands synchronisation of these two steps in semantics. For example, had we given the variable  $x_7$  in place

of  $x_8$ , the result would have been much different.

$$\begin{array}{c}
 \text{every} \quad \text{man} \quad [8 \quad [t_7 \text{ runs}]] \\
 \\
 (5.91) \quad \frac{\frac{\lambda P. \lambda Q. \forall x_0. P(x_0) \rightarrow Q(x_0) \quad \text{man}' \quad \lambda P. \lambda x_8. P \quad x_7 \quad \text{run}'}{\lambda Q. \text{man}'(x_0) \rightarrow Q(x_0)} \quad \frac{\vdots \quad \text{run}'(x_7)}{\lambda x_8. \text{run}'(x_7)}}{\forall x_0. \text{man}'(x_0) \rightarrow \text{run}'(x_7)}
 \end{array}$$

For in the last step we have

$$\begin{aligned}
 & (\lambda Q. \forall x_0. \text{man}'(x_0) \rightarrow Q(x_0))(\lambda x_8. \text{run}'(x_7)) \\
 = & \forall x_0. \text{man}'(x_0) \rightarrow (\lambda x_8. \text{run}'(x_7))(x_0) \\
 = & \forall x_0. \text{man}'(x_0) \rightarrow \text{run}'(x_7)
 \end{aligned}$$

Thus only if the binder abstracts the same variable that the trace denotes do we get the correct quantification. The problems evidently get worse if we have more than one quantifier.

In light of Alphabetical Innocence we can now see why this project is bound to fail. For the meaning of  $[t_8 \text{ run}]$  and  $[t_7 \text{ run}]$  must be the same. Thus, movement has the side effect of displaying the variable. Now, quantifier movement was originally done to obtain alternate scopings (it was used to this effect by Montague, too, though not under that name). The idea was that different readings are the effect of a different structure beyond the level of VP.

(5.92) Every man loves some woman.

(5.93) every man [8 [some woman [7 [ $t_8 t_7$  loves]]]]

(5.94) some woman [7 [every man [8 [ $t_8 t_7$  loves]]]]

The underlying theme in generative grammar has been to make movement be the central device by which different readings are obtained. We can see however that this has nothing to do with movement, only with the order of quantification. For once we have displayed the variables Alphabetical Innocence strikes and we must be in a position to reabstract the correct variable. But how does the quantifier remember which variable it is supposed to bind?

The generativist will point to the indices in the syntactic structure to answer that question. However, we have also said that notational additions such as numbers cannot be part of the syntactic structure. Additionally, as we have just said,



even if the indices are present in the syntax, they have no meaning in the semantics and therefore the idea of exposing and then abstracting a variable cannot work. If we therefore eliminate all numbers the material relevant for interpretation is only this:

(5.95) [every woman [some man [*t t* loves]]]

(5.96) [some man [every woman [*t t* loves]]]

(I hasten to add that even this contains information that the surface string does not show, for example, the number and places of occurrence of traces.) Now, suppose we were to interpret the LF directly. Then we would have to make sure we know (apart from the scopes of the quantifiers) that /every man/ is the subject and /some woman/ is the object. Unfortunately, we lose precisely that information once we decide to move the quantifier. We are lost.

The impasse has been created by thinking that the interpretation of the quantified NP can and must somehow be delayed. What is apparent, however, is that quite to the contrary the quantified NP must be interpreted immediately, upon inserting it into the structure. One way out of the dilemma (not the only one) is to allow the subject to combine first with the verb. Thus, one way to account for the difference in quantifier scope is to assume that the sentence has the following structures.

(5.97) some man [loves every woman]

(5.98) [some man loves] every woman

All that is required is to have two rules of quantification for a transitive verb. One where one binds the subject, and the other where it binds the object.

This may be hard to digest, but it has been observed that in certain constructions we actually do find the subject-verb constituent (see for example [Steedman, 1990]).

(5.99) Some man loves and the children adore every woman.

While generative grammar has insisted that the observed subject-verb constituent is just a constituent containing the object as well, we have rejected such analyses on two grounds. One is that syntax is not allowed to delete material. The other is that the empty material is of no actual help in establishing the correct semantics.

I should emphasize that in the literature on compositionality one rarely finds people taking offense at the use of free variables. The reason is that the issue of compositionality is often confused with offering just any sort of algorithm to compute the right meanings. The Tarskian truth conditions, formulated in terms of sets of assignments as values for propositions, is perfectly intelligible and rigorously formalized. It therefore passes that test. But is it appropriate? Is the set of assignments sending  $x_8$  (as opposed to  $x_7$ ) to some man really the meaning of /man/? Indeed, one of the few advocates of bound variables, Pauline Jacobson, is actually more worried about how variables are properly administrated rather than whether the Tarskian semantics is a proper choice. Similarly, the literature in Categorical Grammar is full of proposals where free variables are used. If I am right, all these approaches are on the wrong track if they make use of variable names as opposed to linking aspects.

## 5.5 The Structure of Dutch

In this section we shall look at arguments in favour of syntactic structure. The previous section already gave a glimpse of the idea that sentence structure can be motivated from purely semantic considerations. In the remainder of the chapter we shall develop this idea further. Traditionally in linguistics, arguments in favour of a particular syntactic structure were backed mostly by syntactic tests (substitution, movement and so on). These tests were surface tests. The tests themselves are based on certain background assumptions. Let us take the example of transformations.

(5.100) It is easy to please John.

(5.101) To please John is easy.

The correlation between (5.100) and (5.101) were taken to show that the sentence (5.100) contains a constituent /to please John/. The argument was that we can apply a movement transformation to (5.100) to get (5.101). As much as this sounds like a reasonable proposal, there is no reason to assume that (5.101) is derived from (5.100). Technically, we just have two different sentences. (Present day transformational grammar actually does *not* derive (5.101) from (5.100).) What makes this argument at all acceptable is the fact that there is not just a syntactic correlation; the transformation would not have been proposed to derive (5.101)

from (5.100) if it had not been for the fact that they mean (approximately) the same thing. Indeed, the idea that gave rise to transformations in the first place was that they can capture meaning correspondences on the basis of syntactic regularities. Even though Chomsky has changed the concept of transformation, the idea that they should not interfere with meaning has been an underlying theme all along. I give two examples that show how semantics is relevant.

There is a systematic syntactic correlation between a transitive sentence and one where subject and object are exchanged (ignoring subject verb agreement):

(5.102) John sees Mary.

(5.103) Mary sees John.

This does not work if one of them is a pronoun for reasons of case; and in other languages it might not work for case reasons. (Making the transformations suitably complex is a way to deal with that problem, however.) Yet in English this correlation is systematic. However, no one proposes a transformation that does this. Similarly, the well known attachment paradoxes do not lead to the proposal of a transformation, to derive, say, (5.106) from (5.105):

(5.104) The police saw a man with a telescope.

(5.105) The police saw [a man with a telescope].

(5.106) The police [[saw a man] with a telescope].

The fact that the interpretation of passive sentences is different from their active counterparts has in fact in the 70s been used to argue against deriving passive from active sentences:<sup>2</sup>

(5.107) Everyone in this class speaks two languages.

(5.108) Two languages are spoken by everyone in this class.

While in (5.108) the universal quantifier has a narrow scope (however only preferentially) (5.107) it has wide scope only.

---

<sup>2</sup>It is a subtle matter to see in what ways such meaning facts can at all bear on the question whether one sentence is derived from another. Because interpretation happens only once in a derivation. The argument would roughly be this. Suppose that meaning is established at the beginning of the derivation (at deep structure). Now suppose that  $S'$  is (more precisely: must be) derived from  $S$  through a transformation. Then the derivation that yields  $S'$  from its deep structure also derives  $S$  on the way. Same deep structure, same meaning. (A dual argument can be used if interpretation is established at LF.) Hence if the two sentences have different meaning they cannot stem from the same deep structure.

It should be clear that the same remarks apply to the use of the substitution method to discover the tree structure of a sentence in a context free language. All these tests assume in one way or another a semantic correlation. It is interesting to note in this connection that the standard understanding of ‘strong generative capacity’ was only the fact that a grammar could generate a language together with the right kind of structure without reference to any semantics. But how do we know that a language has that structure in the first place?

In my view, the answer lies in the fact that these languages are interpreted. The structure turns out to be necessary in order to derive the *interpreted* language not just its string part. We have met arguments of this sort before in Section 3.5. In this section I shall present cases from the literature, some of which have been the cause of intense debate. I shall show that the semantic theory developed in the previous chapter allows us to say something quite nontrivial about the syntactic structure of natural languages.

The first case is that of Dutch infinitives. Here is what they look like.

- (5.109) Ik zeg dat de kinderen zwemmen.  
*I say that the children swim.*
- (5.110) Ik zeg dat Marie de kinderen leert zwemmen.  
*I say that Mary teaches the children to swim.*
- (5.111) Ik zeg dat Piet Marie de kinderen laat leren zwemmen.  
*I say that Piet lets Mary teach the children to swim.*
- (5.112) Ik zeg dat Jan Pier Marie de kinderen ziet laten leren  
zwemmen.  
*I say that Jan sees Piet let Mary teach the children to swim.*

The order in which the elements appear in the Dutch sentences is quite different from English. All the NPs come first, followed by the verbs. Within the verbs we find first a finite verb and then infinitives. Second, the verbs line up in the same way as in English and not in reverse order. Thus we do not have

- (5.113) \*Ik zeg dat Marie de kinderen zwemmen leert.
- (5.114) \*Ik zeg dat Piet Marie de kinderen zwemmen leren laat.
- (5.115) \*Ik zeg dat Jan Pier Marie de kinderen zwemmen leren  
laten zag.

This word order is the order of German. But in Dutch this order is ungrammatical. However the reason it is ungrammatical is only that the finite verb is at the end and the nonraising verb at the beginning. Thus, to make any of the above grammatical, we just have to flip the verbs at either end of the sequence of verbs. But even if we were to do this, we would get grammatical sentences but their meaning would be different from that of the German sentence in that same order. Thus we have to keep in mind that the difference between Dutch and German runs deeper than the surface order would make us believe. It will turn out that under our conception of strong generative capacity Dutch is not strongly context free, but German is. However, Dutch still is weakly context free. Let us see how we can establish this. First notice that the methods of Section 3.5 cannot be directly applied without inquiring into the nature of semantics. The reason is Theorem (3.15). It seems plausible that the construction of Dutch is both unambiguous and monophone. Hence the reason for the impossibility cannot just be combinatorial. It must have to do with the way semantics works. We shall show below what that extra property is. Let us mention here that the claim that Dutch is not weakly context free is originally due to [Huybregts, 1984], which came at a time when Gazdar and Pullum were revisiting arguments by Chomsky and others concerning the trans context freeness of languages. This culminated in the book [Gazdar *et al.*, 1985], which presented an elaborate unification based context free grammar mechanism for natural language. This book provoked the idea that human languages are universally context free, and this is why there was renewed interest in the question. Huybregts was aware of the semantic flavour of his argument, and it took [Shieber, 1985] to get the point home that some languages are non context free after all. What Shieber showed was however that Swiss German (more exactly Züritütsch, the dialect spoken in Zurich) was not even weakly context free. Thus, the argumentation remained strictly confined to form (be it syntax or morphology).

To be able to actually prove some facts about Dutch we are going to simplify and formalize matters somewhat. The simplification consists in ignoring tense, using only singulars, and no finite forms. It is a trivial matter to extend the accounts below to the less simplified case. I trust that the reader has knowledge of a few facts concerning CF languages (see [Harrison, 1978] or [Kracht, 2003]). These are that if  $L \subseteq A^*$  is a CF string language, and  $R \subseteq A^*$  a regular string language, then  $L \cap R$  also is CF. Another is that if  $\varphi : A \rightarrow B^+$  is an arbitrary map and  $L \subseteq A^*$  is CF then  $\varphi[L]$  also is CF. (Notice that  $\varphi(a)$  must be nonempty for all  $a \in A$ !) These techniques are used to infer that the fragment below ‘scales’ up to the full language, that is to say, can be used to infer that Dutch as a whole,

and not just this selected fragment, is not CF. I shall not perform that argument since it essentially requires syntactic arguments (and more empirical facts about Dutch), and we are more interested in the issue of compositionality. But to make the sentences more realistic would be to obscure the problems that occur at a more fundamental level.

I shall in fact present various different formalisations, all leading basically to the same conclusion but different from each other in subtle but crucial respects.

I shall use predicate logic with constants for names and basic predicates. There are two sorts: individuals, and events. To include events is to make the formal semantic account less trivial. It would similarly be possible to use time points or intervals, but events are actually easier to use. The arities of the verbs is different according to their meaning. The base verbs are unary, and the raising verbs take two arguments of each sort. For example,  $\text{let}'(e_0, e_1, x_0, x_1)$  means ‘ $e_0$  is an event of letting, whose subject is  $x_0$ , who is granting  $x_1$  to perform  $e_1$ ’. Since  $x_1$  is then also the subject of the embedded event  $e_1$  ( $x_1$  is said to ‘perform  $e_1$ ’) there is some nontrivial argument identification going on under merge. We shall also assume to have argument roles to further decompose the meanings of the verbs. Thus we actually regard  $\text{let}'(e_0, e_1, x_0, x_1)$  as an abbreviation:

$$(5.116) \quad \text{let}'(e_0, e_1, x_0, x_1) := \\ \text{let}'(e_0) \wedge \text{thm}'(e_0, e_1) \wedge \text{agt}'(e_0, x_0) \wedge \text{ben}'(e_0, x_1) \wedge \text{agt}'(e_1, x_1)$$

The reason for this assumption will soon become apparent.

Thus, in addition to the standard vocabulary, the predicate logic will contain: constants of type  $o$  (‘object’) for each name, constants of type  $e$  for each verb, constants of type  $\langle e, o \rangle$  and  $\langle e, e \rangle$  for argument roles, and identity.

**Example 71.** We now present our first language. Our basic vocabulary is as follows:

$$(5.117) \quad \begin{array}{llll} \langle \text{Piet}, & \langle x_0 = \text{p}' \rangle & \langle \text{zwemmen}, & \langle \text{swim}'(e_0, x_0) \rangle \rangle \\ \langle \text{Jan}, & \langle x_0 = \text{j}' \rangle & \langle \text{let}, & \langle \text{let}'(e_0, e_1, x_0, x_1) \rangle \rangle \\ \langle \text{Marie}, & \langle x_0 = \text{m}' \rangle & \langle \text{leren}, & \langle \text{teach}'(e_0, e_1, x_0, x_1) \rangle \rangle \\ \langle \text{het kind}, & \langle x_0 = \text{c}' \rangle & \langle \text{zien}, & \langle \text{see}'(e_0, e_1, x_0, x_1) \rangle \rangle \end{array}$$

This is to say that the exponents are considered minimal units (if you will, letters of an alphabet), and their meanings are as given. For each of them there is a constant  $f_{\vec{x}}$  with exponent  $\vec{x}$ , and it is interpreted as given above.

We assume that the only constituents are of the form, where  $m = n$  or  $m = n + 1$ :

$$(5.118) \quad \text{NP}_0 \text{—} \text{NP}_1 \text{—} \cdots \text{—} \text{NP}_{n-1} \text{—} \text{V}_0 \text{—} \text{V}_1 \text{—} \cdots \text{—} \text{V}_{m-1}$$

The meaning of such an expression is the one that it ordinarily has in Dutch. If  $n = m$  it is a concept of type  $\langle e, o \rangle$ , involving an event variable and an object variable. If  $n = m + 1$  it is a concept of type  $\langle e, o, o \rangle$ .

First we present a grammar of Dutch that generates this language. Constituents are either strings or pairs of strings. NPs by themselves as well as Vs are strings. All other exponents are analysed as pairs  $\langle \vec{x}, \vec{y} \rangle$  where  $\vec{x}$  is a sequence of NPs and  $\vec{y}$  a sequence of Vs. Thus they have the form (5.118). We shall use two functions: one integrates a verb, and the second an NP.

We start with the base case. Let  $c \otimes d$  be defined as follows. (a) It is partial and requires that  $c$  is a 1-concept of type  $\langle o \rangle$  and  $d$  a 2-concept of type  $\langle e, o \rangle$ , that is, it is a function of an object and an event; (b) the result is obtained by identifying the object of  $c$  with that of  $d$ . Since there is only one of each sort, we do not even need a linking aspect for this to be well-defined.

$$(5.119) \quad \mathcal{I}(c)(\langle \vec{x}, c \rangle, \langle \vec{y}, d \rangle) := \begin{cases} \langle \langle \vec{x}, \text{—} \vec{y} \rangle, c \otimes d \rangle & \text{if } \vec{x} \text{ is an NP and } \vec{y} \text{ a nonraising} \\ & \text{verb.} \\ \text{undefined} & \text{else.} \end{cases}$$

Now we deal with the recursion in the construction.

Say that a pair  $\langle \vec{x}, \vec{z} \rangle$  is of **Type A** if  $\vec{x}$  is a sequence of  $n$  NPs and  $\vec{z}$  a sequence of  $n$  Vs, and  $n > 0$ .

$$(5.120) \quad \mathcal{I}(v)(\langle \langle \vec{x}, \vec{z} \rangle, c \rangle, \langle \vec{y}, d \rangle) := \begin{cases} \langle \langle \vec{x}, \vec{z} \text{—} \vec{y} \rangle, c \otimes' d \rangle & \text{if } \langle \vec{x}, \vec{z} \rangle \text{ is of Type A and } \vec{z} \text{ a raising verb.} \\ \text{undefined} & \text{else.} \end{cases}$$

Here,  $c \otimes' d$  is defined if and only if  $c$  is of type  $\langle e, o \rangle$  and  $d$  of type  $\langle e, e, o, o \rangle$ . It identifies the event variable of  $c$  with the second event variable of  $d$ , and the object variable of  $c$  with the second object variable of  $d$ ; then it quantifies the event variable away. To do this, we need to have a linking aspect that defines the notions 'first' and 'second' for concepts denoted by raising verbs in the appropriate way.

This can be done by simply listing the critical sets for each of the raising verbs. The other strategy is semantic. We choose a linking aspect for  $\text{thm}'$  (since this is of type  $\langle e, e \rangle$ ). This allows to distinguish first and second event variable. For the object variables we actually take advantage of the thematic predicates  $\text{agt}'$  (giving us the first variable) and  $\text{ben}'$  (giving us the second).

Thus we get the following meaning of (English) ‘let Mary swim’:

$$(5.121) \quad \langle \text{let}'(e_0, e_1, x_0, x_1) \rangle \otimes' \langle \text{swim}'(e_0, x_0) \wedge x_0 = \text{p}' \rangle \\ = \langle \exists e_1. \text{let}'(e_0, e_1, x_0, x_1) \wedge \text{swim}'(e_1, x_1) \wedge x_0 = \text{p}' \rangle$$

The last function needed is the one that incorporates the NP.  $\langle \vec{x}, \vec{z} \rangle$  is of **Type B** if it is a sequence of  $n$  NPs followed by  $n + 1$  Vs. Define a function  $\otimes''$  as follows. It is defined if and only if  $c$  is of type  $\langle e, o, o \rangle$  and  $d$  of type  $\langle o \rangle$ . It identifies the object of  $d$  with the second object of  $c$  and the quantifies that away. Notice that we can define first and second object using the thematic predicate  $\text{agt}'$  (picking out the ‘first’ argument). This will be the meaning of (English) ‘Piet let Mary swim’:

$$(5.122) \quad \langle \exists e_1. \text{let}'(e_0, e_1, x_0, x_1) \wedge \text{swim}'(e_0, x_0) \wedge x_0 = \text{m}' \rangle \otimes'' \langle x_0 = \text{p}' \rangle \\ = \langle \exists x_1. \exists e_1. \text{let}'(e_0, e_1, x_0, x_1) \wedge \text{swim}'(e_1, x_1) \wedge x_0 = \text{p}' \wedge x_1 = \text{m}' \rangle$$

With this definition we put

$$(5.123) \quad \mathcal{J}(n)(\langle \langle \vec{x}, \vec{z} \rangle, c \rangle, \langle \vec{y}, d \rangle) \\ := \begin{cases} \langle \langle \vec{y}, \vec{x}, \vec{z} \rangle, c \otimes'' d \rangle & \text{if } \langle \vec{x}, \vec{z} \rangle \text{ is of Type B and } \vec{z} \text{ an NP.} \\ \text{undefined} & \text{else.} \end{cases}$$

Let us now see why a context free grammar for this language cannot be given. Let us take a look at the sentence we just derived:

$$(5.124) \quad \text{Jan\_Marie\_Piet\_laten\_leren\_zwemmen}$$

In line with the assumptions that strings must contain the same number of NPs and Vs or at most one more V than NP, we can only propose the following parts (in addition to the words themselves):

$$(5.125) \quad \begin{array}{l} \text{Jan\_Marie\_Piet\_laten\_leren\_zwemmen,} \\ \text{Marie\_Piet\_laten\_leren\_zwemmen,} \\ \text{Piet\_laten\_leren,} \\ \text{Piet\_laten} \end{array}$$



Figure 5.2: A Derivation

$$\begin{aligned}
& \mathcal{U}(nf_{Jan} \vee n f_{Marie} \vee c f_{Piet} f_{zemma} f_{leren} f_{laten}) \\
= & \mathcal{J}(n)(\langle \text{Jan}, \langle x_0 = j' \rangle \rangle, \mathcal{J}(v)(\mathcal{J}(n)(\langle \text{Marie}, \langle x_0 = m' \rangle \rangle, \mathcal{J}(v)(\mathcal{J}(c)(\langle \text{Piet}, \langle x_0 = p' \rangle \rangle, \\
& \langle \text{zemma}, \langle \text{swim}'(e_0, x_0) \rangle \rangle), \langle \text{leren}, \langle \text{teach}'(e_0, e_1, x_0, x_1) \rangle \rangle)) \\
& \langle \text{laten}, \langle \text{let}'(e_0, e_1, x_0, x_1) \rangle \rangle)) \\
= & \mathcal{J}(n)(\langle \text{Jan}, \langle x_0 = j' \rangle \rangle, \mathcal{J}(v)(\mathcal{J}(n)(\langle \text{Marie}, \langle x_0 = m' \rangle \rangle, \mathcal{J}(v)(\langle \langle \text{Piet}, \perp \text{zemma} \rangle, \\
& \langle \text{swim}'(e_0, x_0) \wedge x_0 = p' \rangle \rangle), \langle \text{leren}, \langle \text{teach}'(e_0, e_1, x_0, x_1) \rangle \rangle)) \\
& \langle \text{laten}, \langle \text{let}'(e_0, e_1, x_0, x_1) \rangle \rangle)) \\
= & \mathcal{J}(n)(\langle \text{Jan}, \langle x_0 = j' \rangle \rangle, \mathcal{J}(v)(\mathcal{J}(n)(\langle \text{Marie}, \langle x_0 = m' \rangle \rangle, \langle \langle \text{Piet}, \perp \text{leren} \perp \text{zemma} \rangle, \\
& \langle \exists e_1. \text{swim}'(e_1, x_1) \wedge x_1 = p' \wedge \text{teach}'(e_0, e_1, x_0, x_1) \rangle \rangle), \\
& \langle \text{laten}, \langle \text{let}'(e_0, e_1, x_0, x_1) \rangle \rangle)) \\
= & \mathcal{J}(n)(\langle \text{Jan}, \langle x_0 = j' \rangle \rangle, \mathcal{J}(v)(\langle \langle \text{Marie} \perp \text{Piet}, \perp \text{leren} \perp \text{zemma} \rangle, \\
& \langle \exists x_1. \exists e_1. \text{swim}'(e_1, x_1) \wedge \text{teach}'(e_0, e_1, x_0, x_1) \wedge x_0 = m' \wedge x_1 = p' \rangle \rangle), \\
& \langle \text{laten}, \langle \text{let}'(e_0, e_1, x_0, x_1) \rangle \rangle)) \\
= & \mathcal{J}(n)(\langle \text{Jan}, \langle x_0 = j' \rangle \rangle, \mathcal{J}(v)(\langle \langle \text{Marie} \perp \text{Piet}, \perp \text{leren} \perp \text{zemma} \rangle, \\
& \langle \exists e_1. \text{swim}'(e_1, p') \wedge \text{teach}'(e_0, e_1, x_0, p') \wedge x_0 = m' \rangle \rangle), \\
& \langle \text{laten}, \langle \text{let}'(e_0, e_1, x_0, x_1) \rangle \rangle)) \\
= & \mathcal{J}(n)(\langle \text{Jan}, \langle x_0 = j' \rangle \rangle, \langle \langle \text{Marie} \perp \text{Piet}, \text{laten} \perp \text{leren} \perp \text{zemma} \rangle, \\
& \langle \exists e_0. \exists e_1. \text{swim}'(e_1, p') \wedge \text{teach}'(e_0, e_1, x_0, p') \wedge x_0 = m' \wedge \text{let}'(e_2, e_0, x_2, x_0) \rangle \rangle)) \\
= & \langle \langle \text{Jan} \perp \text{Marie} \perp \text{Piet}, \text{laten} \perp \text{leren} \perp \text{zemma} \rangle, \\
& \langle \exists x_0. \exists e_1. \text{swim}'(e_1, p') \wedge \text{teach}'(e_0, e_1, x_0, p') \wedge x_0 = m' \wedge \text{let}'(e_2, e_0, x_2, x_0) \\
& \wedge x_2 = j' \rangle \rangle)) \\
= & \langle \langle \text{Jan} \perp \text{Marie} \perp \text{Piet}, \text{laten} \perp \text{leren} \perp \text{zemma} \rangle, \\
& \langle \exists e_0. \exists e_1. \text{swim}'(e_1, p') \wedge \text{teach}'(e_0, e_1, m', p') \wedge \text{let}'(e_2, e_0, x_2, m') \wedge x_2 = j' \rangle \rangle))
\end{aligned}$$

In this case we are done: only the first two strings contain a raising verb. It is easy to see that this argument works in the general case, too.  $\odot$

This example worked because we had fixed the language to be in a certain way. Whether or not it is that way, is an empirical issue. Linguists have had serious difficulties assessing the nature of the constituents in the sentences above (from a syntactic viewpoint). If we make the choice as above, there is not much chance for a CFG. Yet, one may complain that we have been biased: coordination facts indicate, for example, that the verb sequences can be constituents, too (see [Groenink, 1997]), and we have just excluded them. Therefore, we shall now ease the constituency of Dutch somewhat by admitting more subconstituents. There is

another point where we might have made an arbitrary decision. The meaning of a sentence or complex expression is a function of the meanings of its parts. We have admitted this function to do only the following:

- ① identify some columns (= add an identity of the form  $x_i = x_j$ ), and
- ② cylindrify (= apply an existential quantifier  $\exists x_i$ ).

There does not seem to be much room for choices when to apply ①. After all, identifying two variables is to say something significant. On the other hand, applying ② seems to be negotiable from a meaning point of view. The difference between various choices seems to be rather of technical nature. When a variable has been quantified away it is not available any more for identification. On the other hand, the more free variables we have the more difficult the job of identifying the right one gets.

**Example 72.** We shall extend the set of meaningful constituents to include all strings of NPs followed by Vs which are substrings of sentences. This means, effectively, that all sequences of names and verbs are licit which contain at most one nonraising V, and where the NPs precede the Vs and the raising Vs precede the nonraising Vs. This, by the way, is a regular language. As interpretation we choose the one induced by these strings as parts of some sentence. In each combination of a V  $\vec{x}$  and a V  $\vec{y}$  following it, we shall identify the theme of  $\vec{x}$  with the event nontheme of  $\vec{y}$ ; we shall also identify the benefactor of  $\vec{x}$  with the agent of  $\vec{y}$ . No existential quantification. This is a variant of  $\textcircled{\Delta}$ '' above. With respect to the NPs, matters are different. Consider the string /Jan.Piet.leren/. Is Jan the one who teaches? It depends. For the string could be embedded in the following different sentences:

(5.126) Jan Piet leren zwemmen

(5.127) Marie Jan Piet leren laten zwemmen

In (5.126), Jan is doing the teaching, and Piet the swimming. In (5.127), Jan is not doing the teaching, it is Marie. However, if Jan is doing the teaching, Piet is the one who is being taught. (This is because they are adjacent, and in Dutch the next NP is the beneficiary of the action carried out by the agent.) Thus, we assume that

our language contains the following signs:

(5.128)  $\langle \text{Jan\_Piet\_leren}, \langle \text{teach}'(e_0, e_1, x_0, x_1) \wedge x_2 = j' \wedge x_3 = p' \rangle \rangle$

(5.129)  $\langle \text{Jan\_Piet\_leren}, \langle \text{teach}'(e_0, e_1, x_0, x_1) \wedge x_0 = j' \wedge x_1 = p' \rangle \rangle$

The more NPs we have in our string, the more signs we seem to get in this way. However, there are some more restrictions. The verb following the rightmost NP is certainly the highest. So in the following example we cannot make Piet the beneficiary of the teaching. Still, three signs remain:

(5.130)  $\langle \text{Marie\_Jan\_Piet\_leren}, \langle \text{teach}'(e_0, e_1, x_0, x_1) \wedge x_2 = m' \wedge x_3 = j' \wedge x_4 = p' \rangle \rangle$

(5.131)  $\langle \text{Marie\_Jan\_Piet\_leren}, \langle \text{teach}'(e_0, e_1, x_0, x_1) \wedge x_0 = m' \wedge x_1 = j' \wedge x_2 = p' \rangle \rangle$

$\langle \text{Marie\_Jan\_Piet\_leren}, \langle \text{teach}'(e_0, e_1, x_0, x_1) \wedge x_1 = m' \wedge x_2 = j' \wedge x_3 = p' \rangle \rangle$

To show this, look at the following sentences containing them.

(5.132) Marie Jan Piet leren laten leren laten zwemmen

(5.133) Marie Jan Piet leren laten leren zwemmen

(5.134) Marie Jan Piet leren laten zwemmen

And so, with  $n$  NPs and 1 V we have  $n$  choices in general. Notice, however, that if the last V is nonraising, the number of different readings is just 1. This is because the subject of the nonraising verb must be the last NP, and the subject of the verb before it the second last NP, and so on.

The only exception to this is when the string does not contain an NP. This case deserves some attention. In the case of raising verbs we need to take care of two event variables and two object variables. Each verb clearly identifies an order between its variables. Let the first verb introduce  $e_0$  and  $e_1$  and the second  $e_2$  and  $e_3$ . Then we have to identify  $e_1$  and  $e_2$ ; after that we can quantify away  $e_1/e_2$ . The complex concept has only two free event variables. On the other hand, we do not really need to quantify any variable. The concept establishes an order between the three variables ( $e_0$ ,  $e_1$  and  $e_3$ ). For example, in /leren laten/ we have to combine  $\langle \text{let}'(e_0, e_1, x_0, x_1) \rangle$  with  $\langle \text{teach}'(e_0, e_1, x_0, x_1) \rangle$ . Let us rename the variables in the second formula and return to ordinary predicates:

(5.135)  $\text{let}'(e_0, e_1, x_0, x_1) \wedge \text{teach}'(e_2, e_3, x_2, x_3)$


The result we want is (up to renaming)

$$(5.136) \quad \text{let}'(e_0, e_1, x_0, x_1) \wedge \text{teach}'(e_1, e_3, x_1, x_3)$$

Furthermore, given that we can identify a linear order on the event variables it is also possible to define a linear order on the object variables. This is because we can identify via the thematic roles which of the variables is actually the agent (beneficiary) of which event variable. In this way the newly formed concept can be effectively merged with any new concept. The effect is that the constituency in the verb cluster is completely free.

Let us see how we can derive the meanings of the sentences using these signs. In view of the last remark it appears that there is no other choice but to start by assembling the entire V cluster. For suppose we did not do that. Then we build signs of the form NP sequence followed by V sequence. These are multiply ambiguous, yet only one of the readings is the one needed in the sentence. It is just that as long as we do not have the last V, we do not know which one we should choose. Now, if we do not make a choice then we simply postpone the choice. However, if we do that we discard information about the relative order of the NPs (since this is not recorded in the semantics, only in the string). Thus the requirement we get is this: the NP cluster is right branching, while the V cluster has any structure we please. The easiest structure (but not the only one) is a right branching structure:

$$(5.137) \quad [\text{Jan} [\text{Piet} [\text{Marie} [\text{het kind} [\text{zien} [\text{laten} [\text{leren} \\ \text{zwemmen}]]]]]]]]$$

Once again, however, Dutch is not context free. To see this one must appeal to Ogden's Lemma. Instead, I shall just point out that since the verb clusters each form a constituent, there must be infinitely many categories (one for each number of Vs). 

I conclude this discussion with the following remarks. The structure is in basic agreement with CCG. It has indeed been proposed that the structure of Dutch involves a verbal cluster. [Groenink, 1997] has also argued from coordination data that the verbal cluster is more flexible in Dutch and German.

## 5.6 Arguing for Syntactic Structure

The previous section has shown that Dutch (or at least some ‘purified’ version thereof) is indeed not weakly context free. The book [Gazdar *et al.*, 1985] seems to have shown, however, that at least English is CF. Many syntactic theories seem to agree on this (see [Rogers, 1994] and [Kracht, 1995] for a demonstration that generative grammar of the 80s was saying precisely this). In this section we shall look at some constructions of English that indicate that also English is not CF.

(5.138) John, Mary and Phil sang, danced, and played drums,  
respectively.

This sentence is to be interpreted as follows: it is a conjunction of “John sang”, “Mary danced” and “Phil played the drums”. Without the word /*respectively*/ it could be interpreted as saying that John, Mary and Phil *each* sang, danced *and* played drums.

(5.139) John, Mary and Phil sang, danced, and played drums.

The interpretation of (5.139) requires only a basic sentential structure: we have a plural NP /John, Mary and Phil/ and a VP /sang, danced and played drums/. Each has a coordinated structure. However, (5.138) is much different. To make the argumentation self-contained we consider the following data.

**Example 73.** The language contains the following signs (compare the grammars  $P_1$  to  $P_3$  of Section 3.2). We choose a domain  $U$  of individuals. Intransitive verbs and nouns denote sets of individuals. There are  $n$  intransitive verbs  $v_i$ ,  $i < n$ , and  $2^n$  nouns. Verb forms are in the past, so that number distinctions do not exist. For every combination of  $v_i$  (or their negation) we assume that there is exactly one name  $n_j$  such that  $n_j$  satisfies that combination. The legitimate strings are defined by  $S$  (where  $V$  denotes any verb,  $N$  any name):

$$\begin{aligned}
 Y & := (N \cdot \_ )^+ \text{and} \_ \cdot N \\
 (5.140) \quad Z & := (V \cdot \_ )^+ \text{and} \_ \cdot V \\
 S & := Y \cup Z \cup N \cup V \cup Y \cdot \_ \cdot Z \cdot (, \_ \text{respectively})? \cdot .
 \end{aligned}$$

Additionally we assume that if /*respectively*/ is present, the number of names and the number of verbs is the same. This defines a context free language (we

leave the proof as an exercise). What we shall show here however is that no compositional CFG exists.

The interpretations are as following. (a) Strings from  $Y$  denote the sets of all denotations of occurring names, (b) strings from  $Z$  denote the intersection of all the sets denoted by the individual members; (c) strings from  $YZ$  denote the intersection of what  $Y$  denotes and what  $Z$  denotes; (d) finally, let  $\vec{y}_i, i < n + 1$  be some names and  $\vec{z}_i, i < n + 1$ , some verbs. Then the denotation of

(5.141)  $\vec{y}_0 \dots \vec{y}_{n-1}$  and  $\vec{z}_0 \dots \vec{z}_{n-1}$  and  $\dots$  respectively.

is the intersection of the denotations of  $/\vec{x}_i \vec{y}_i./$  for all  $i < n + 1$ .

Let us see what happens if we attempt to interpret (5.138) using the same structure as for (5.139). In that case the following happens. The phrase */John, Mary and Phil/* is synonymous with */John, Phil and Mary/* and also */Mary, John and Phil/* and so on. However, this synonymy does not exist between (5.138) and (5.142) and (5.143).

(5.142) John, Phil and Mary sang, danced, and played drums,  
respectively.

(5.143) John, Phil and Mary sang, danced, and played drums,  
respectively.

It follows that we cannot assume that */John, Mary and Phil/* is a constituent in (5.138). Similarly we argue that neither */John, Mary/* nor */John, Phil/* nor */Mary and Phil/* can be a constituent. And we can do the same with the verbs. The only constituents that we *can* form without running a risk of conflation are */John sang/*, */Mary danced/* and */Phil played drums/*.

It follows that in a construction involving */respectively/* we are forced to assume what is known as **crossover (crossing) dependencies**.

(5.144)  $NP_0 NP_1 \dots NP_{n-1} VP_0 NP_1 \dots$  and  $VP_{n-1}$  respectively.

We can assume that we get these structures as follows. One method is to assume that exponents are pairs of strings  $\langle \vec{x}, \vec{v} \rangle$  such that  $\vec{x}$  is an NP and  $\vec{v}$  an agreeing VP. Let **Case A** be the following property.

(5.145) **Case A** :  $\vec{v}$  does not end with */respectively/*

Furthermore, let  $\otimes$  be the “obvious” conjunction of concepts. Assuming that NPs and VPs denote sets of individuals,  $\otimes$  is intersection of its minimal member, accompanied by existential closure (thus we get a 0-ary concept, also known as a truth value). For two 0-ary concepts,  $\otimes$  is set intersection. (If that presents difficulties, you may replace concepts with standard relations.)

$$(5.146) \quad r(\langle\langle\vec{x}, \vec{u}\rangle, m\rangle, \langle\langle\vec{y}, \vec{v}\rangle, n\rangle) \\ := \begin{cases} \langle\langle\vec{x} \sqcap \vec{y}, \vec{u} \sqcap \vec{v}, \text{and} \sqcap \vec{v}, \text{respectively.}\rangle, m \otimes n\rangle & \text{Case A} \\ \langle\langle\vec{x} \sqcap \vec{y}, \vec{u} \sqcap \vec{v}\rangle, m \otimes n\rangle & \text{else} \end{cases}$$

This makes  $\text{NP}_i$  and  $\text{VP}_i$  in (5.144) into a constituent, which we form as follows.

$$(5.147) \quad s(\langle\vec{x}, m\rangle, \langle\vec{u}, n\rangle) := \begin{cases} \langle\langle\vec{x}, \vec{u}\rangle, \mathbf{C}_0.m \otimes n\rangle & \text{if } \vec{x} \text{ is an NP and } \vec{u} \text{ a VP} \\ \text{undefined} & \text{else} \end{cases}$$

Another is to assume that the NP-VP constituents are not even formed. In that case we use a modified version of  $r$ :

$$(5.148) \quad r^*(\langle\langle\vec{x}, m_0\rangle, \langle\vec{u}, m_1\rangle, \langle\vec{y}, n_0\rangle, \langle\vec{v}, n_1\rangle) \\ := \begin{cases} \langle\langle\vec{x} \sqcap \vec{y}, \vec{u} \sqcap \vec{v}, \text{and} \sqcap \vec{v}, \text{respectively.}\rangle, \\ \quad \mathbf{C}_0.(m_0 \otimes m_1) \otimes \mathbf{C}_0.(n_0 \otimes n_1)\rangle & \text{Case A} \\ \text{undefined} & \text{else} \end{cases} \\ r^{**}(\langle\langle\vec{x}, m_0\rangle, \langle\langle\vec{u}, m_1\rangle, \langle\vec{y}, \vec{v}\rangle, n_0\rangle) := \\ \begin{cases} \langle\langle\vec{x} \sqcap \vec{y}, \vec{u} \sqcap \vec{v}\rangle, \mathbf{C}_0.(m_0 \otimes m_1) \otimes n_0\rangle & \text{not Case A} \\ \text{undefined} & \text{else} \end{cases}$$

The first variant is more elegant. ⊗

**Intermission 3.** The grammar and interpretation of sequences of NPs is interesting in its own right.

(5.149) John, Paul and Mary

(5.150) John, Paul or Mary

Assume that coordination requires the presence of either /and/ or /or/. Assume further that meanings are concepts. Finally, the interpretation of a name is assumed to be a singleton set. There are then two choices for us. We can either

interpret the coordinated NP as a relation between the named people, or as the set of all of them. Either of them satisfy the basic laws of conjunction and disjunction (commutativity, associativity and idempotence). It is clear that the overall structure of a conjunction is not unique, even with all this being given. It is trivial to observe that we could in principle design ternary rules, for example. Or we may use wrapping. But we should not dismiss any of these options either, despite the fact that they are more complicated than the obvious right regular grammar.

In a compositional grammar this has noteworthy consequences. If one wishes to make */John and Mary/* a subconstituent of a sentence, then this can only be done if either */Mary and John/* cannot be substituted for it or else the resulting sentence has the same meaning. If you choose to have categories, one can of course discriminate a number of different coordinations, for example, by giving */John and Mary/* a different category than */Mary and John/*. Apart from being rather unsatisfactory, the Principle of the Equality of Indiscernibles (see Page 56) rules this out as well. (It does not under certain circumstances, however. One is agreement in languages where a conjunct controls the same agreement as its last member. Latin is such a case. In such circumstances, since */John and Mary/* controls feminine agreement and */Mary and John/* masculine agreement, they have different category.) ☹

Notice that */respectively/* has more syntactic possibilities than given in this example. The preceding argument assumes that we are forming a compositional grammar. Alternatively, and interestingly, even if one does not assume compositionality, the result follows. This has to do with the fact that we have restricted the semantic functions.

English provides yet another construction that is quite problematic in phrase structure terms, namely *gapping*. This phenomenon is illustrated in the following sentence.

(5.151) John gave Mary a kiss and George Susan a flower.

We understand this as the conjunction of two sentences:

(5.152) John gave Mary a kiss.

(5.153) George gave Susan a flower.

What is problematic about this construction is that it forces us to assume that we have a discontinuous constituent */John Mary a flower/*. Let us see why this



is so. Like the previous example, we assume that the meaning of sentences is a truth value. (That assumption can of course be modified, though the argument would not work as easily.) Suppose, we first fully compose the sentence (5.152). This will have as its meaning, say, a truth value. In this case it is impossible to interpolate the meaning of the verb so that it can be used to derive the meaning of (5.153). For notice that rather than having the full (5.153) we are actually missing the verb. It follows that (5.151) does not contain the constituent (5.152)!

Instead we are led to assume that (5.151) contains the constituents /John Mary a kiss/ and /George Susan a flower/. More precisely, it contains the pairs  $\langle \text{John, Mary a kiss} \rangle$  and  $\langle \text{George, Susan a flower} \rangle$ . The verb /gave/ is inserted into both of them. Since gapping is like conjunction in allowing any number of parts, we propose a solution similar to the one offered for respectively.

**Example 74.** Here is a sketch of gapping. The constituents of the form /George Susan a flower/ are seen as pairs  $\langle \text{George, Susan a flower} \rangle$ . These pairs are coordinated via the mode  $c$ . After all of them are coordinated, the verb is linked with the conjunctive meaning and inserted between the first subject and the first object.

$$(5.154) \quad \mathcal{J}(c)(\langle \langle \vec{x}, \vec{y} \rangle, m \rangle, \langle \langle \vec{u}, \vec{v} \rangle, n \rangle) := \langle \langle \vec{x}, \vec{y} \vec{u} \vec{v}, \vec{v}, m \cup n \rangle$$

$$(5.155) \quad \mathcal{J}(i)(\langle \langle \vec{x}, \vec{y} \rangle, m \rangle, \langle \vec{v}, n \rangle) := \langle \vec{x} \vec{v} \vec{y}, m \oplus^3 n \rangle$$

This accounts for this type of gapping. ⊛

It may seem to be disappointing that the syntactic structures are so irregular. Syntactic theories sometimes give the impression that syntactic structure (at least of English) is a matter of a few universal principles. This seems to be an artefact of the data that the theories wish to explain. No one theory succeeds in giving us a satisfactory account of all known phenomena and typically they tend to do well in a few areas and badly in others. I should also point out that in the literature there are no essentially different solutions to the problems shown above. Respectively-constructions have been used in [Kac *et al.*, 1987] to show that English is not context free. Where the latter authors use the distribution of pronouns to show that the string language of English is not context free, here we have used of the meanings to derive the same conclusion.

**Exercise 65.** Write a CFG to generate  $S$  from Example 73.



# Chapter 6

## Conclusion

In this book I have tried to build a theory that lets us ask (and answer) questions concerning the structure of languages. Some of the results plainly validate some of our intuitions; others have been surprising (at least to me). The road has been fairly difficult not the least because exact results are difficult to obtain, and because new techniques had to be found.

We are now at the end of our journey. Many questions have been answered, and many new ones arose. I shall summarise this work with a few remarks.

- ☆ There are tangible results that have been established. For example, it has been established that it is not possible to reduce all ambiguous languages to unambiguous ones (at least if we want to keep the syntactic complexity). Or that concept based predicate logic with infinitely many variables does not have a compositional context free grammar. These results seem to be pretty robust. They cannot be made to disappear if minor changes are made to the languages.
- ☆ The study of interpreted languages really has just begun. We need to understand better in what ways the shift from string languages to interpreted languages changes our outlook on various issues. Mathematically, new combinatorial methods need to be developed. They might help us to understand better in what ways semantics determines syntactic structure.
- ☆ On the way I have tried to make progress also concerning the overall struc-

ture of language. For example, notions such as morphological transparency, realphabetisation, and abstraction were attempts at understanding why natural language apparently has more structure (in the sense of architecture in terms of levels or strata) than the present frameworks (and others) make believe.

- ☆ Negative results are typically hard to obtain. This contrasts with a lot of claims in the literature that suggest that certain phenomena force us to adopt or abandon a specific framework because of compositionality. Most of these results either follow because quite specific assumptions are made at the outset or because the authors simply are not imaginative enough about counterstrategies. For example, I have not been able to show conclusively that there is no TAG for boolean logic if we allow the semantic functions to be partial, though it seems certain that this claim is true.
- ☆ The results established here make use of some additional hypotheses about language some of which are indispensable such as the hypothesis that rules do not destroy any structure. Others might be more controversial, for example that syntactic structures are sequences of strings and nothing else.
- ☆ The literature in formal semantics operates with high powered tools. Often however the justification in using them is only that they provide a functioning algorithm without clarifying whether or not that algorithm deserves the label 'compositional'. Our approach has been not to rely on particular mechanisms but rather to clarify identity criteria of meaning (such as alphabetic innocence) and see how much follows from them.

# Appendix A

## Useful Mathematical Concepts and Notation

For a set  $S$  we write  $\text{card } S$  for the cardinality of  $S$  (which is to say the number of elements of  $S$ ). A number  $n$  is the set of all numbers  $i$  (including 0) such that  $i < n$ . Thus,  $3 = \{0, 1, 2\}$ . (The interested reader may check that therefore  $0 = \emptyset$ ,  $1 = \{0\} = \{\emptyset\}$ ,  $2 = \{\emptyset, \{\emptyset\}\}$ , and  $3 = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}$ .) Thus,  $i < n$  and  $i \in n$  are synonymous. Writing  $f : k \rightarrow n$  means that  $f$  is a function defined on all numbers  $< k$ , with values  $< n$ .

We shall write  $\langle x_0, x_1, \dots, x_{n-1} \rangle$  for the tuple of length  $n$  consisting of  $x_0, x_1$ , etc., in that order. We make no commitment about the real nature of tuples; you may think of them as functions from the set  $n$  to the domain. (In that case they are the same as strings.) The length of  $\vec{x} := \langle x_0, \dots, x_{n-1} \rangle$  is denoted by  $|\vec{x}|$ . We write  $x_0$  in place of  $\langle x_0 \rangle$  even though they are technically distinct. Tuple formation is not associative. So,  $\langle x_0, \langle x_1, x_2 \rangle \rangle$  is not the same as  $\langle \langle x_0, x_1 \rangle, x_2 \rangle$ . If  $\vec{x} = \langle x_0, \dots, x_{m-1} \rangle$  and  $\vec{y} = \langle y_0, \dots, y_{n-1} \rangle$  are tuples, the concatenation is denoted as follows.

$$(A.1) \quad \vec{x} \cdot \vec{y} := \langle x_0, \dots, x_{m-1}, y_0, \dots, y_{n-1} \rangle$$

Repetitions are not eliminated, so this is a sequence of length  $m + n$ .

Given two sets,  $A$  and  $B$ ,  $A \times B$  is the set of pairs  $\langle a, b \rangle$  such that  $a \in A, b \in B$ . Given an indexed family  $A_i, i \in I$ , of sets,  $\prod_{i \in I} A_i$  is the set of functions from  $I$  to the union of the  $A_i$  such that  $f(i) \in A_i$  for all  $i \in I$ . (Thus, technically,  $A_0 \times A_2$  is *not* the same as  $\prod_{i \in 2} A_i$ , though the difference hardly matters.) Let  $A$  and  $B$  be

sets. A **relation from  $A$  to  $B$**  is a subset of  $A \times B$ . We write  $x R y$  in place of  $\langle x, y \rangle \in R$ . A **partial function from  $A$  to  $B$**  is a relation from  $A$  to  $B$  such that  $x R y$  and  $x R z$  implies  $y = z$ . A **function from  $A$  to  $B$**  is a partial function from  $A$  to  $B$  where for every  $x \in A$  there is a  $y \in B$  such that  $x R y$ . We write  $f : A \rightarrow B$  to say that  $f$  is a function from  $A$  to  $B$  and  $f : A \hookrightarrow B$  to say that  $f$  is a partial function from  $A$  to  $B$ . If  $f : A \rightarrow B$  and  $g : B \rightarrow C$  then  $g \circ f : A \rightarrow C$  is defined by  $(g \circ f)(x) := g(f(x))$ . We write  $\text{dom}(f)$  for the set of all  $a \in A$  such that  $f$  is defined on  $a$ . If  $f : A^n \hookrightarrow B$  and  $S \subseteq A$  then we write  $f \upharpoonright S$  for the following function

$$(A.2) \quad (f \upharpoonright S)(\vec{x}) := \begin{cases} f(\vec{x}) & \text{if } \vec{x} \in S^n \text{ and } f(\vec{x}) \text{ is defined} \\ \text{undefined} & \text{else} \end{cases}$$

A somewhat simpler definition is

$$(A.3) \quad f \upharpoonright S := f \cap S^n \times A$$

If  $X \subseteq A$  is a set we write  $f[X] := \{f(a) : a \in X, a \in \text{dom}(f)\}$ . This is the **direct image of  $X$  under  $f$** . In particular,  $\text{rng}(f) := f[A]$  is the **range** of  $f$ .  $f$  is **surjective** or **onto** if  $\text{rng}(f) = B$ .  $f$  is **injective** or **into** if for all  $x, y$ : if  $f(x)$  and  $f(y)$  are defined then either  $x = y$  or  $f(x) \neq f(y)$ . A **permutation** is a surjective function  $f : n \rightarrow n$ . It is easily seen that if  $f$  is surjective it is also injective. There are  $n! := n(n-1)(n-2) \cdots 2 \cdot 1$  permutations of an  $n$  element set.

When  $f : A \times B \rightarrow C$  is a function, we say that it is **independent** of  $A$  if for all  $x, x' \in A$  and  $y \in B$ ,  $f(x, y) = f(x', y)$ . Pick  $x \in A$  and define  $\hat{f} : B \rightarrow C$  by  $\hat{f}(y) := f(x, y)$ . If  $f$  is independent of  $A$ ,  $\hat{f}$  is independent of the choice of  $x$ . For partial functions there are some subtleties. We say that  $f$  is **weakly independent of  $A$**  if for all  $x, x' \in A$  and  $y \in B$ , if  $f(x, y)$  and  $f(x', y)$  exist, they are equal.  $f$  is **strongly independent of  $A$**  if for all  $x, x' \in A$  and  $y \in B$ , if  $f(x, y)$  exists then so does  $f(x', y)$  and they are equal. By default, for partial functions we say that it is independent of  $A$  if it is weakly independent. Independence of  $B$  is defined similarly. Similarly, if  $f$  has several arguments, it may be weakly or strongly independent of any of them.

If  $f : A \rightarrow C$  and  $g : A \rightarrow D$  are functions, then  $f \times g : x \mapsto \langle f(x), g(x) \rangle$  is a function from  $A$  to  $C \times D$ . Every function from  $A$  to  $C \times D$  can be decomposed into two functions, in the following way. Let  $\pi_C : \langle x, y \rangle \mapsto x$  and  $\pi_D : \langle x, y \rangle \mapsto y$  be the projections from  $C \times D$  to  $C$  and  $D$ , respectively. Then we have the general

equation

$$(A.4) \quad f = (\pi_C \circ f) \times (\pi_D \circ f)$$

and so the functions  $\pi_C \circ f$  and  $\pi_D \circ f$  are the decomposition. This picture changes when we turn to partial functions. From a pair  $f : A \hookrightarrow C$  and  $g : A \hookrightarrow D$  we can form the partial function

$$(A.5) \quad (f \times g)(x) := \begin{cases} \langle f(x), g(x) \rangle & \text{if both } f(x) \text{ and } g(x) \text{ are defined} \\ \text{undefined} & \text{else} \end{cases}$$

Unfortunately,  $f \times g$  does not allow to recover  $f$  and  $g$  uniquely. The problem is this: we have

$$(A.6) \quad \text{dom}(f \times g) = \text{dom}(f) \cap \text{dom}(g)$$

However, from an intersection it is not easy to recover the individual sets. If  $A = \{0\}$ ,  $f = \{(0, c)\}$  and  $g = \emptyset$  (the empty partial function) then  $f \times g = \emptyset$ . However, also  $\emptyset \times \emptyset = \emptyset$ .

If  $n$  is a number a bijective function  $f : n \rightarrow n$  is called a **permutation of  $n$** .  $\Pi_n$  denotes the set of all permutations of  $n$ . Permutations are most conveniently described using the following notation. Pick a number  $i < n$ . The **cycle** of  $i$  is the largest sequence of the form  $i, f(i), f(f(i)), \dots$  in which no member is repeated. The set  $\{i, f(i), f^2(i), \dots\}$  is also called the **orbit** of  $i$  under  $f$ . We write this cycle in the form  $(if(i)f(f(i)) \dots f^{k-1}(i))$ . An example is  $(2567)$ , which says that  $f$  maps 2 to 5, 5 to 6, 6 to 7 and 7 to 2. The **order of the cycle** is  $k$ . It is not hard to see that  $f^k(i) = i$ . For if  $f^k(i) = f^m(i)$  for some  $m < k$  then also  $f^{k+1}(i) = f^{m+1}(i)$  (since  $f$  is a function), and  $f^{k-1}(i) = f^{m-1}(i)$  (since  $f$  is bijective, so its inverse is a function, too). It follows that  $f^{k-m}(i) = i$ , and since  $m < k$ , we must have  $m = 0$ . (Else we have found a number  $j > 0$  smaller than  $k$  such that  $f^j(i) = i$ .) Cycles can be cyclically rotated: for example,  $(2567) = (5672)$ . It is easy to see that any two distinct orbits are disjoint. A permutation thus partitions the set  $n$  into orbits, and defines a unique cycle on each of the orbits. In writing down permutations, cycles of length 1 are omitted. Cycles permute and can be cyclically rotated. Thus we write  $(2567)(3)(1)(04)$  and  $(2567)(04), (5672)(40)(3), (04)(2567)$  interchangeably. The permutation that changes nothing is also denoted by  $()$ .

A **group** is a structure  $\mathcal{G} = \langle G, 1, ^{-1}, \cdot \rangle$ , where  $1 \in G$ ,  $^{-1} : G \rightarrow G$  and  $\cdot : G \times G \rightarrow G$  are such that for all  $x, y, z \in G$ :

1.  $1 \cdot x = x \cdot 1 = x$ .
2.  $x^{-1} \cdot x = x \cdot x^{-1} = 1$ .
3.  $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ .

We say that  $x^{-1}$  is the **inverse** of  $x$ , and that  $x \cdot y$  is the **product** of  $x$  and  $y$ . The set  $\Pi_n$  forms a group. The product is defined by  $(f \cdot g)(x) := f(g(x))$ . The unit is the permutation  $()$ . The inverse is obtained as follows. The inverse of a cycle  $(i_0 i_1 \cdots i_{k-1})$  is the cycle  $(i_{k-1} i_{k-2} \cdots i_1 i_0)$ . The inverse of a series of disjoint cycles is obtained by inverting every cycle individually. (Note that if  $c$  and  $d$  are disjoint cycles, then  $c \cdot d = d \cdot c$ .) A **subgroup** of  $\mathcal{G}$  is a triple  $\mathcal{H} = \langle H, 1^*, {}^{-1*}, \cdot^* \rangle$  where  $H \subseteq G$ ,  $1^* = 1$ ,  $x^{-1*} = x^{-1}$  and  $x \cdot^* y = x \cdot y$ . It is stated without proof that if  $\mathcal{H}$  is a subgroup of  $\mathcal{G}$  then  $|H|$  divides  $|G|$ .

A **signature** is a pair  $\langle F, \Omega \rangle$  (often written simply  $\Omega$ ) where  $F$  is a set (the set of **function symbols**) and  $\Omega : F \rightarrow \mathbb{N}$  a function, assigning each function symbol an arity. An  **$\Omega$ -algebra** is a pair  $\mathfrak{A} = \langle A, I \rangle$  such that for every  $f \in F$ ,  $I(f) : A^{\Omega(f)} \rightarrow A$ . We also write  $f^{\mathfrak{A}}$  for  $I(f)$ . A **partial  $\Omega$ -algebra** is a pair  $\mathfrak{A} = \langle A, I \rangle$  where for each  $f \in F$ ,  $I(f) : A^{\Omega(f)} \hookrightarrow A$ . A **weak congruence** on  $\mathfrak{A}$  is an equivalence relation  $\Theta \subseteq A^2$  such that the following holds.

If  $a_i \Theta b_i$  for every  $i < \Omega(f)$  and both  $I(f)(a_0, \dots, a_{\Omega(f)-1})$  and  $I(f)(b_0, \dots, b_{\Omega(f)-1})$  exist then they are equal.

$\Theta$  is **strong** if whenever  $a_i \Theta b_i$  for all  $i < \Omega(f)$  then  $I(f)(a_0, \dots, a_{\Omega(f)-1})$  exists iff  $I(f)(b_0, \dots, b_{\Omega(f)-1})$  exists as well. If  $\Theta$  is a strong congruence we can construct the so-called **quotient algebra**  $\mathfrak{A}/\Theta$ .

$$\begin{aligned}
 (A.7) \quad & a/\Theta := \{b : a \Theta b\} \\
 & A/\Theta := \{a/\Theta : a \in A\} \\
 & (I/\Theta)(f)(a_0/\Theta, \dots, a_{\Omega(f)-1}/\Theta) := (f(a_0, \dots, a_{\Omega(f)-1}))/\Theta \\
 & \mathfrak{A}/\Theta := \langle A/\Theta, I/\Theta \rangle
 \end{aligned}$$

It is to be observed that  $(I/\Theta)(f)$  is well defined; the value of the function does not depend on the choice of representatives. Moreover, whether or not it is defined is also independent of the choice of representatives, since the congruence is strong.



If  $\mathfrak{A} = \langle A, I \rangle$  and  $\mathfrak{C} = \langle C, J \rangle$  are partial algebras then the **product** of  $\mathfrak{A}$  and  $\mathfrak{C}$  is defined by

$$(A.8) \quad (I \times J)(f)(\langle a_0, c_0 \rangle, \dots, \langle a_{\Omega(f)-1}, c_{\Omega(f)-1} \rangle) := \langle I(f)(\vec{a}), J(f)(\vec{c}) \rangle$$

We write  $\mathfrak{A} \times \mathfrak{C}$  for the product.

In the domain of algebra, the term functions and polynomial functions are very important. Their definition is notoriously difficult since one is often required to use variables where this creates problems due to choices of alphabetical variants. Instead, I offer the following definition, which only uses functions and compositions.

- ① All projections  $p_i^n : A^n \rightarrow A$  defined by  $p_i^n(a_0, \dots, a_{n-1}) := a_i$  are term functions.
- ② If  $g_i : A^{m_i} \rightarrow A$ ,  $i \in \Omega(f)$ , are term functions and  $p := \sum_{i \in \Omega(f)} m_i$ , then  $f \circ \langle g_0, \dots, g_{\Omega(f)-1} \rangle : A^p \rightarrow A$  is a term function where

$$f \circ \langle g_0, \dots, g_{\Omega(f)-1} \rangle(\vec{c}_0, \dots, \vec{c}_{\Omega(f)-1}) := f(g_0(\vec{c}_0), \dots, g_{\Omega(f)-1}(\vec{c}_{\Omega(f)-1}))$$

is a term function.

- ③ If  $g : A^n \rightarrow A$  is a term function and  $i < j$  then  $g \circ \Delta_{ij}^n : A^{n-1} \rightarrow A$  defined by  $(g \circ \Delta_{ij}^n)(a_0, \dots, a_{n-2}) := g(a_0, \dots, a_{j-1}, a_i, a_j, a_{j+1}, \dots, a_{n-1})$  also is a term function.

(For a partial algebra, replace “function” everywhere by “partial function”). Term functions are often described by means of terms such as  $(x + y) \cdot z$ , but this is inaccurate. A **polynomial** is defined to a term function over the expanded algebra  $\mathfrak{A}^A$ , where for each  $a \in A$  we have added a constant  $\underline{a}$  to the language, whose interpretation is fixed to  $a$ . (Alternatively, it is the closure under ① – ③ of the set of functions containing  $A^0 \rightarrow A : \emptyset \rightarrow a$  for each  $a$ .)

# Symbols

$\vec{x}, \vec{x} \vec{y}, / \cdot /$ , 20	$f^\varepsilon, f^\mu$ , 87
$A^*, A^+$ , 20	$f \star g$ , 88
$S \mid T, S \cdot T, ST, S^n, S^*, S^+$ , 20	$\mathcal{J}^\varepsilon, \mathcal{J}^\mu$ , 88
:digit:, 22	$G^\times, G_\times$ , 88
$\Omega$ , 25	$f_*^\mu$ , 95
$\mathbb{N}$ , 25	$\geq$ , 95
:eq:, 27	$f_*^\varepsilon$ , 97
$\text{Tm}_\Omega(V)$ , 28	$H(\gamma)$ , 108
$\iota_G(t)$ , 29	$G_\times$ , 109
$L(G)$ , 30	$e^\circ$ , 123
:bool:, 31	$L^\S$ , 123
:blet:, 34	$e^\vee, L^\vee$ , 129
$\iota_G(\cdot)(\bar{s})$ , 40	Bool, 131
$[\vec{x}/x]$ , 41	$L \dot{\vdash} B$ , 134
$\sim_G [\cdot \cdot \cdot]_G$ , 44	$\sharp_a(\cdot)$ , 134
$\varepsilon(\cdot), \kappa(\cdot)$ , 45	$G \dot{\vdash} D$ , 134
$\varepsilon[\cdot], \kappa[\cdot]$ , 45	
$\vec{u} \Rightarrow_R \vec{v}, \vec{u} \Rightarrow_R^n \vec{v}$ , 46	$M_\alpha$ , 154
$A \vdash_G \vec{x}$ , 46	$A_>, A_<$ , 154
$L(G)$ , 47	$M_{\vec{s}}$ , 160
$[A]_G$ , 47	$\Xi$ , 160
$G^\diamond$ , 48	$\beta$ , 161
$L^c(G)$ , 50	$\sim_v$ , 161
$p^{A^*}$ , 54	$[\cdot]_{\mathcal{M}}$ , 161
$G^b$ , 63	fr( $\cdot$ ), 162
occ( $\vec{y}, t$ ), 65	$\ell(\cdot), R^{\rightarrow k}$ , 163
	$(\cdot)$ , 163
$\varepsilon(\cdot), \mu(\cdot)$ , 86	$\mathbb{C}_i$ , 163
$\varepsilon[\cdot], \mu[\cdot]$ , 86	$\pi[\cdot]$ , 165

$E(\cdot)$ , 165  
 $P_t(\cdot)$ , 165  
 $\llbracket \cdot \rrbracket$ , 167  
 $\dagger, \bar{\dagger}$ , 167  
 $\ell(\cdot)$ , 168  
 $\S(\cdot \cdot \cdot)$ , 170  
 $c \leq d$ , 170  
 $\leftrightarrow$ , 172  
 $\otimes^{f,g}$ , 176  
 $\binom{L}{2}$ , 181  
 $L^+$ , 181  
 $\delta(\mathcal{R}), \delta(\mathcal{C})$ , 197  
 $f^Y(c)$ , 197  
 $\rho_R\{\vec{p}\}$ , 198  
  
 $L_\tau$ , 206  
 $\triangleleft$ , 212  
 $\text{CL}_\tau^n$ , 213  
 $\text{tp}(\chi)$ , 213  
  
 $\text{card}$ , 253  
 $|\vec{x}|$ , 253  
 $\vec{x} \cdot \vec{y}$ , 253  
 $\bigtimes_{i \in I} A_i$ , 253  
 $f \upharpoonright S$ , 254  
 $\mathfrak{A}/\Theta$ , 256  
 $\mathfrak{A} \times \mathfrak{C}$ , 257

# Index

- a-term, 22
- abstraction, 145
  - equivalent, 146
- additivity, 134
- adjacency, 181
- adjunction rule
  - string, 58
- algebra, 256
  - partial, 256
- allophone, 145
- alphabet, 20
- ambiguity
  - lexical, 123
  - spurious, 123
  - structural, 123
- analysis, 31
- arity, 25
- assignment, 161
- autonomy, 108
  
- bigrammar, 88
  - balanced, 89
  
- c-grammar, 108
- c-language, 108
- c-sign, 108
- c-string, 44
- categorial autonomy, 108
- category, 44, 107
- CFG
  - left regular, 118
  - right regular, 118
- compositionality, 108
  - direct, 83
  - rule-to-rule, 83
- concatenation grammar, 54
- concept, 167
  - type, 170
- congruence
  - strong, 256
  - weak, 256
- connectivity property, 55
- constant, 25
- context, 35, 58
- context free grammar, 45
- converse, 166
- coordinate frame, 182
- crossover dependencies, 246
- cycle, 255
  - order, 255
  
- degree of embedding, 136
- denotation, 86
- deprofiling, 196
- depth
  - embedding, 220
- derivability, 258
- derivation, 46, 47, 52
- distance, 182
- duality, 97

- expansion, 165
  - diagonal, 165
  - generalised diagonal, 166
  - product, 166
- exponent, 86
- expression
  - comple, 80
  - simple, 80
- expressive power, 86
  
- falsum concept, 167
- first degree equivalence, 144
- formula, 129, 160
  - atomic, 160
- formula atomic, 206
- fragment, 134
- function, 254
  - partial, 254
  - polynomial, 41
  - term, 41
  
- generation, 59
- grammar, 25
  - ambiguous, 31
  - autonomous, 97
  - c-string, 45
  - concatenation, 54
  - context free, 45, 56
  - extensional independent, 97
  - extensionally autonomous, 97
  - extensionally compositional, 95
  - independent, 97
  - interpreted, 87
  - language, 47
  - language of, 26, 58
  - primitive, 69
  - semiautonomous, 97
  - semicompositional, 95
  - standard, 66
  - string adjunction, 59
  - syntactically well regimented, 104
  - transparent, 65
  - unambiguous, 31
- group, 255
  
- homology, 197
- homomorphism, 24
  
- image
  - direct, 254
- independence, 108, 254
  - strong, 254
  - weak, 254
- indeterminacy
  - semantically spurious, 130
- indeterminate grammar, 58
- index, 129, 206
- interpreted grammar
  - autonomous, 130
  - compositional, 130
  - indeterminate, 130
  - language of, 130
- inverse, 256
  
- language, 20
  - abstract, 192
  - autonomous, 97
  - c-string, 45
  - compositional, 82
  - context free (CF), 47
  - grounding, 192
  - independent, 97
  - interpreted, 86
  - interpreted compositional, 95
  - monophone, 87
  - narrow sense, 30
  - string, 86

- strongly  $\mathcal{C}$ , 118
- strongly context free, 118
- superstrongly  $\mathcal{C}$ , 118
- superstrongly context free, 118
- transparent, 65
- unambiguous, 87
- wide sense, 30
- langue, 146
- lexicon, 26, 58
- line, 182
- linking aspect, 177
- locale, 58
- location, 181
- main symbol, 136
- meaning, 86
- mode, 25, 58
  - lexical, 26
  - nonlexical, 26
- model, 161
- morphological transparency, 86
- necessity, 210
- object
  - realisation, 183
- object schema, 183
- occurrence, 35
  - accidental, 65
  - constituent, 65
  - syncategorematic, 66
- opposition, 144
- orbit, 255
- parole, 146
- part, 33, 35
- permutation, 165, 254, 255
- phases, 222
- phone, 143
- phoneme, 145
- picture, 183
- pivot, 178
- polynomial, 41, 257
  - linear string, 54
  - string, 41
- possibility, 210
- product, 256, 257
- pseudoadditivity, 134
- quotient algebra, 256
- range, 254
- realphabetisation, 24
- relata, 143
- relation, 254
- reprofiling, 197
- rule, 26, 58
- sampler, 228
- set
  - critical, 177
  - deductively closed, 162
  - definable, 173
- signature, 25, 256
  - first-order, 160
- signifié, 87
- signifiant, 87
- signs, 86
- sort, 159
- space, 181
  - connected, 181
- space of signs, 87
- string, 20
  - ambiguous, 31
  - empty, 20
  - length, 20
  - ungrammatical, 31
- X-string, 52

- structure, 161
  - canonical, 210
- subalgebra, 107
- subgroup, 256
- subterm, 29
- symbol
  - relation, 160
- syntactic object, 163
  - complete, 163
- syntax
  - abstract, 25
  - concrete, 25
- term, 27
  - $G$ -, 28
  - analysis, 31
  - categorially equivalent, 70
  - constant, 28
  - definite, 89
  - indefinite, 89
  - intersubstitutable, 70
  - orthographically definite, 29, 89
  - semantically definite, 89
- theory, 162
  - consistent, 162
  - maximally consistent, 210
- tmesis, 73
- trigrammar, 109
- truth, 161
- type, 213
  - functional, 164
  - relation, 160
  - relational, 159
- typed object, 159
- ua-term, 22
- unfolding, 29
- utterance, 143
- valuation, 161, 206
- variable, 206
- variant, 166
  - extensional, 94
  - immediate, 166
- verum concept, 167
- world, 210





# Bibliography

- [Barker and Jacobson, 2007] Chris Barker and Pauline Jacobson, editors. *Direct Compositionality*. Number 14 in Oxford Studies in Theoretical Linguistics. Oxford University Press, Oxford, 2007.
- [Ben Shalom, 1996] Dorit Ben Shalom. *Semantic Trees*. PhD thesis, Department of Linguistics, UCLA, 1996.
- [Benaceraff, 1973] Paul Benaceraff. Mathematical Truth. *Journal of Philosophy*, 70:661–679, 1973.
- [Bittner, 2006] Maria Bittner. Online Update. Temporal, Modal and de se Anaphora in Polysynthetic languages. In Chris Barker and Pauline Jacobson, editors, *Direct Compositionality*, pages 363–404. Oxford University Press, Oxford, 2006.
- [Chomsky, 1986] Noam Chomsky. *Barriers*. MIT Press, Cambridge (Mass.), 1986.
- [Chomsky, 1993] Noam Chomsky. A Minimalist Program for Linguistic Theory. In K. Hale and S. J. Keyser, editors, *The View from Building 20: Essays in Honour Sylvain Bromberger*, pages 1–52. MIT Press, 1993.
- [Copestake *et al.*, 2005] Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. Minimal Recursion Semantics: An Introduction. *Research on Language and Computation*, 3:281–332, 2005.
- [Dixon, 1994] R. M. W. Dixon. *Ergativity*. Number 69 in Cambridge Studies in Linguistics. Cambridge University Press, Cambridge, 1994.

- [Dorr, 2004] Cian Dorr. Non-symmetric relations. In Dean W. Zimmerman, editor, *Studies in Metaphysics, Vol. 1*, pages 155–192. Oxford University Press, 2004.
- [Erdélyi Szabó *et al.*, 2007] Miklós Erdélyi Szabó, László Kálmán, and Ági Kurucz. Towards a natural language semantics without functors and operands. *Journal of Logic, Language and Information*, 16, 2007.
- [Falk, 2001] Yehuda Falk. *Lexical-Functional Grammar: An Introduction to Parallel Constraint-Based Syntax*. CSLI, Stanford, 2001.
- [Fiengo and May, 2006] Robert Fiengo and Robert May. *De lingua belief*. MIT Press, Cambridge (Mass.), 2006.
- [Fine, 2000] Kit Fine. Neutral relations. *The Philosophical Review*, 109:1–33, 2000.
- [Fine, 2003] Kit Fine. The Role of Variables. *Journal of Philosophy*, 50:605–631, 2003.
- [Fine, 2007] Kit Fine. *Semantic relationism*. Blackwell, London, 2007.
- [Gärdenfors, 2004] Peter Gärdenfors. *Conceptual Spaces*. MIT, Cambridge (Mass.), 2004.
- [Gazdar *et al.*, 1985] Gerald Gazdar, Ewan Klein, Geoffrey Pullum, and Ivan Sag. *Generalized Phrase Structure Grammar*. Blackwell, London, 1985.
- [Groenink, 1997] Annius Groenink. *Surface without Structure. Word Order and Tractability Issues in Natural Language Analysis*. PhD thesis, University of Utrecht, 1997.
- [Harrison, 1978] Michael A. Harrison. *Introduction to Formal Language Theory*. Addison Wesley, Reading (Mass.), 1978.
- [Heim and Kratzer, 1998] Irene Heim and Angelika Kratzer. *Semantics in Generative Grammar*. Blackwell Publishers, Oxford, 1998.
- [Hodges, 2001] Wilfrid Hodges. Formal features of compositionality. *Journal of Logic, Language and Information*, 10:7–28, 2001.

- [Huybregts, 1984] R. Huybregts. Overlapping Dependencies in Dutch. *Utrecht Working Papers in Linguistics*, 1:3–40, 1984.
- [IPA, 1999] IPA. *Handbook of the International Phonetic Association*. Cambridge University Press, Cambridge, 1999.
- [Jacobson, 1999] Pauline Jacobson. Towards a variable free semantics. *Linguistics and Philosophy*, 22:117–184, 1999.
- [Jacobson, 2000] Pauline Jacobson. Paycheck pronouns, Bach-Peters sentences, and variable free semantics. *Natural Language Semantics*, 8:77–155, 2000.
- [Jacobson, 2002] Pauline Jacobson. The (Dis)Organisation of the Grammar: 25 Years. *Linguistics and Philosophy*, 25:601–626, 2002.
- [Janssen, 1997] Theo Janssen. Compositionality. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 417–473. Elsevier, Amsterdam, 1997.
- [Kac *et al.*, 1987] Michael B. Kac, Alexis Manaster-Ramer, and William C. Rounds. Simultaneous–Distributive Coordination and Context–Freeness. *Computational Linguistics*, 13:25–30, 1987.
- [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.
- [Kamp, 1981] Hans Kamp. A theory of truth and semantic representation. In Jeroen Groenendijk, editor, *Formal methods in the study of language*. Mathematisch Centrum, 1981.
- [Keenan and Stabler, 2001] Edward L. Keenan and Edward P. Stabler. *Bare Grammar. Lectures on Linguistics Invariants*. CSLI, Stanford, 2001.
- [King, 2007] Jeffrey C. King. *The nature and structure of content*. Oxford University Press, Oxford, 2007.
- [Kornai, 2007] András Kornai. *Mathematical Linguistics. Advanced Information and Knowledge Processing*. Springer, Berlin, 2007.
- [Korpela, 2006] Jukka Korpela. *Unicode Explained*. O’Reilly, Sebastopol, CA, 2006.

- [Kracht, 1995] Marcus Kracht. Syntactic Codes and Grammar Refinement. *Journal of Logic, Language and Information*, pages 41–60, 1995.
- [Kracht, 2003] Marcus Kracht. *Mathematics of Language*. Mouton de Gruyter, Berlin, 2003.
- [Kracht, 2006] Marcus Kracht. Partial Algebras, Meaning Categories and Algebraization. *Theoretical Computer Science*, 354:131–141, 2006.
- [Kracht, 2007] Marcus Kracht. Is Adjunction Compositional? to appear, 2007.
- [Lamb, 1966] Sydney M. Lamb. *Outline of Stratificational Grammar*. Georgetown University Press, Washington, 1966.
- [Landmann, 2004] Fred Landmann. *Indefinites and the Type of Sets*. Number 3 in Explorations in Semantics. Blackwell, Oxford, 2004.
- [Lasersohn, 2006] Peter Lasersohn. Compositional Interpretation in Which the Meaning of Complex Expressions are not Computable from the Meanings of their Parts. Manuscript, 2006.
- [Manaster-Ramer and Michalove, 2001] Alexis Manaster-Ramer and Peter Michalove. Etymology vs. phonology: the treatment of \*/w/ after sonorants in Armenian. *Münchener Studien zur Sprachwissenschaft*, 61:149–162, 2001.
- [Manaster-Ramer *et al.*, 1992] Alexis Manaster-Ramer, M. Andrew Moshier, and R. Suzanne Zeitman. An Extension of Ogden’s Lemma. Manuscript. Wayne State University, 1992.
- [Manaster-Ramer, 1986] Alexis Manaster-Ramer. Copying in natural languages, context-freeness and queue grammars. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, pages 85–89, 1986.
- [Martín-Vide and Păun, 1998] Carlos Martín-Vide and Gheorghe Păun. Structured Contextual Grammars. *Grammars*, 1:33–55, 1998.
- [Matthews, 1978] P. H. Matthews. *Inflectional morphology. An introduction to the theory of word-structure*. Cambridge Textbooks in Linguistics. Cambridge University Press, 1978.
- [Mel’cuk, 1993 2000] Igor Mel’cuk. *Cours de Morphologie Générale*, volume 1 – 5. Les Presses de l’Université de Montréal, 1993 – 2000.

- [Mel'čuk, 1988] Igor Mel'čuk. *Dependency Syntax: Theory and Practice*. SUNY Linguistics Series. State University of New York Press, Albany, 1988.
- [Miller, 1999] Philip H. Miller. *Strong Generative Capacity. The Semantics of Linguistic Formalisms*. CSLI, Stanford, 1999.
- [Monk, 1976] Donald J. Monk. *Mathematical Logic*. Springer, Berlin, Heidelberg, 1976.
- [Onions, 1973] C. T. Onions. *The Shorter English Dictionary*. Oxford University Press, Oxford, 1973.
- [Parsons, 1994] Terence Parsons. *Events in the Semantics of English. A Study in Subatomic Semantics*. Number 19 in Current Studies in Linguistics. MIT Press, 1994.
- [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.
- [Pullum and Rawlins, 2007] Geoffrey Pullum and Kyle Rawlins. Argument or no argument? *Linguistics and Philosophy*, 30:277–287, 2007.
- [Putnam, 1975] Hilary Putnam. The meaning of 'meaning'. In *Mind, Language and Reality*, pages 215–271. Cambridge University Press, Cambridge, 1975.
- [Radzinski, 1990] Daniel Radzinski. Unbounded Syntactic Copying in Mandarin Chinese. *Linguistics and Philosophy*, 13:113–127, 1990.
- [Rautenberg, 2006] Wolfgang Rautenberg. *A Concise Introduction to Mathematical Logic*. Springer, Berlin, 2006.
- [Rogers, 1994] James Rogers. *Studies in the Logic of Trees with Applications to Grammar Formalisms*. PhD thesis, University of Delaware, Department of Computer & Information Sciences, 1994.
- [Saussure, 1967] Ferdinand de Saussure. *Grundfragen der allgemeinen Sprachwissenschaft*. Walter de Gruyter, Berlin, 2 edition, 1967.

- [Scollon and Wong Scollon, 2003] Ron Scollon and Suzie Wong Scollon. *Discourses in Place. Language in the Material World*. Routledge, London and New York, 2003.
- [Shieber, 1985] Stuart Shieber. Evidence against the Context–Freeness of Natural Languages. *Linguistics and Philosophy*, 8:333–343, 1985.
- [Staudacher, 1987] Peter Staudacher. Zur Semantik indefiniter Nominalphrasen. In Brigitte Asbach-Schnithker and Johannes Roggenhofer, editors, *Neuere Forschungen zur Wortbildung und Historiographie der Linguistik. Festgabe für Herbert E. Brekle*, pages 239–258. Gunter Narr Verlag, Tübingen, 1987.
- [Steedman, 1990] Mark Steedman. Gapping as constituent coordination. *Linguistics and Philosophy*, 13:207–263, 1990.
- [Sternefeld, 2006] Wolfgang Sternefeld. *Syntax. Eine morphologisch motivierte generative Beschreibung des Deutschen*. Stauffenberg Verlag, Tübingen, 2006. 2 Vols.
- [Svenonius, 2007] Peter Svenonius. Paradigm generation and Northern Sámi stems. In Asaf Bachrach and Andrew Nevins, editors, *The Basis of Inflectional Identity*. Oxford University Press, Oxford, 2007.
- [Szabó, 2000] Zoltán Gendler Szabó. Compositionality as supervenience. *Linguistics & Philosophy*, 23:475–505, 2000.
- [Talmy, 2000] Leonard Talmy. *Toward a Cognitive Semantics*, volume 1 & 2. MIT Press, Cambridge (Massachusetts), 2000.
- [Thue, 1914] Axel Thue. Probleme über Veränderungen von Zeichenreihen nach gegebenen Regeln. (Problems concerning changing strings according to given rules). *Skrifter utgit av Videnskapsselskapet i Kristiania, I. Matematisk–naturvidenskabelig klasse*, 10, 1914.
- [Tomalin, 2006] Marcus Tomalin. *Linguistics and the Formal Sciences. The Origins of Generative Grammar*. Cambridge Studies in Linguistics. Cambridge University Press, Cambridge, 2006.
- [Vermeulen, 1995] Kees F. M. Vermeulen. Merging without mystery or: Variables in dynamic semantics. *Journal of Philosophical Logic*, 24:405–450, 1995.

- [Wechsler, 1995] Stephen Wechsler. *The Semantic Basis of Argument Structure*. Dissertations in Linguistics. CSLI Publications, Stanford, 1995.
- [Zadrozny, 1994] Wlodek Zadrozny. From Compositional Semantics to Systematic Semantics. *Linguistics and Philosophy*, 17:329–342, 1994.
- [Zeevat, 1989] Henk Zeevat. A compositional approach to Discourse Representation Theory. *Linguistics and Philosophy*, 12:95–131, 1989.