Chapter 3
Compositionality

THE principle of compositionality is 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.

> 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 from these parts.

Almost every word of this definition is in need of explanation. We begin with the subject of the sentence: *the meaning of a complex expression*. To use this expression here means to acknowledge 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 somewhat differently in Section 2.6; there we concluded that expressions are *sequences* of strings. Moreover, they must be sequences of strings of which we know what their meaning is. This is implicit in the use of the definite determiner in "the meaning of an expression". The use of the definite determiner 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 given* 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;

M. Kracht, *Interpreted Languages and Compositionality*, Studies in Linguistics and Philosophy 89, DOI 10.1007/978-94-007-2108-1_3,
© Springer Science+Business Media B.V. 2011

a language consists in a relation between the two. This is the original idea laid out in Saussure (2011).

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 are given a grammar G of the expressions. An expression is G-**simple** if it is the value in G of a simple term; and an expression is G-**complex** if it is the value in G of a complex term. Often, we omit mentioning the grammar. 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 assume a grammar of English where compounding is performed by concatenation. 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. We wish to say, for example, that idioms are simple. For although as expressions they are complex, their meaning is not derived from the meanings that any proper parts 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 σ 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 is thus 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 from these signs.

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 σ 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. (To be more exact, idioms are signs whose exponent has another meaning together with which it forms a complex sign. The definition of *idiom* is a truly delicate affair.) So, the definition begins by assuming that we have a grammar G and a sign σ. Furthermore, we assume that there is a term function $p(\vec{x})$ and signs $\sigma, \sigma_0, \cdots, \sigma_{n-1}$ such that

$$\sigma = t(\sigma_0, \cdots, \sigma_{n-1}). \tag{3.1}$$

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

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

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

$$m = t^\mu(m_0, \cdots, m_{n-1}). \tag{3.3}$$

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

$$e = f^\varepsilon(\sigma_0, \cdots, \sigma_{\Omega(f)-1}). \tag{3.4}$$

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

$$L = \{\langle \mathsf{a}, 0\rangle, \langle \mathsf{b}, 1\rangle, \langle \mathsf{ab}, 2\rangle, \langle \mathsf{ab}, 3\rangle\} \tag{3.5}$$

Assume that /ab/ is to be considered complex. If we understand a mode to be a syntactic function then this language cannot be compositional, for there is only one function to compose /a/ and /b/.[1] 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:

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

This is a different function but on the strings of the language it shows no difference to plain concatenation. We say therefore that f and g *count as the same mode* exactly when $f^\varepsilon(\vec{\sigma}) = g^\varepsilon(\vec{\sigma})$. (Recall in this connection Example 2.2. The plural of

[1] Well, there are two: $f(x, y) := x^\frown y$ and $g(x, y) := y^\frown x$. But this can be handled by constructing a more complex example.

regular nouns and the third singular of regular verbs are for these purposes formed by the same mode, assuming their arity to be the same.) There are languages that satisfy compositionality even with this strict identity of modes; many computer languages are of that form. There is simply 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 constructionally ambiguous and that accounts for many meaning differences.

Let us now settle down on the final definition of compositionality (see the extended discussion in Section 3.3):

> A language L is *compositional* if there is a grammar G based on a signature $\langle F, \Omega\rangle$ such that (i) $L = L(G)$ and (ii) for each $f \in F$ there is a function f^{μ} such that if $\sigma = \langle e, m\rangle$ and $\sigma_i = \langle e_i, m_i\rangle$, $i < \Omega(f)$, are signs such that
>
> $$\sigma = f(\sigma_0, \cdots, \sigma_{\Omega(f)-1}) \tag{3.7}$$
>
> and g counts as the same mode as f then
>
> $$m = g^{\mu}(m_0, \cdots, m_{\Omega(f)-1}). \tag{3.8}$$

Notice that from (3.7) we deduce that

$$m = f^{\mu}(m_0, \cdots, m_{\Omega(f)-1}), \tag{3.9}$$

since f is the same as f. However, there could be more modes that are the same as f. 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 this 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 this 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 this 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 arguments by Pagin (2003) go in the same direction, though his reasons are slightly different. Pagin argues that speakers and hearers must be able to effectively find meanings associated with expressions and conversely and it is hard to imagine how that can be done without some kind of structural similarity. The structure in meaning is language independent, so this would among others imply a certain similarity between all human languages. I have chosen not to go that way. One reason for my 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^{μ} 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 crane'_b—or a type of machines—call the meaning crane'_m. The meaning function now associates with it the concept $\mathsf{crane}'_b \vee \mathsf{crane}'_m$, which is true of x if and only if x is either a bird crane or a machine crane. The meaning 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 $\{\mathsf{crane}'_b, \mathsf{crane}'_m\}$. 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 individually. Thus we are already imposing a structure onto semantics (that meanings are sets) that languages cannot override. Everything 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 this 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.

__ is square free or it is a product of two prime numbers and greater than 100. (3.10)

This description can be read in two ways. It says that the number is greater than 100, and 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 given above an example of a language that figures in Zadrozny (1994) to show that there are languages that 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 in more than one letter". Rather, it is taken to mean: consists in more than one identifier, where identifiers are sequences of letters not interrupted by special symbols. More complex criteria can be imagined; 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. It

allows tokenisation to precede syntactic analysis. We cannot likewise assume human languages to be built this way. The said property, that complexity is decidable on the basis of the string alone, is called **morphological transparency**. Human languages are therefore in general 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 for example be 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

$$\varepsilon(\sigma) := e, \qquad \mu(\sigma) := m. \tag{3.11}$$

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

$$\varepsilon[L] := \{e : \text{there is } m \in M : \langle e, m \rangle \in L\} \tag{3.12}$$

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

$$\mu[L] := \{m : \text{there is } e \in E : \langle e, m \rangle \in L\} \tag{3.13}$$

the **expressive power of** L.

The meaning of σ is not to be confused with its **denotation**, a term that I wish to avoid since it is often used in a purely extensional sense, while meaning is intensional.

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 (2011). (Chomsky also endorsed that view in Chomsky (1993), though the exponents in Generative Grammar are far more complex.) De Saussure uses the words **signifier** (**signifiant**) and **signified** (**signifié**), rather than *exponent* and *meaning*. The straightforward generalization 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 $G = \langle \Omega, \mathcal{I} \rangle$ where Ω is a finite signature and $\mathcal{I}$ a function that assigns to a symbol $f \in F$ a partial $\Omega(f)$-ary function on $E \times M$:

$$\mathcal{I}(f) : (E \times M)^{\Omega(f)} \hookrightarrow (E \times M). \tag{3.14}$$

Furthermore,

$$L(G) := \{\iota(t) : t \in \mathrm{Tm}_\Omega(\varnothing)\} \tag{3.15}$$

is the **language generated by** G.

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, $\mathcal{I}(f)$ is a partial n-ary function on S. Indeed, the definitions of the previous chapter can be imported without much adaptation. The only difference is that where we generated strings (or sequences thereof) now we generate signs.

I remark here that we can always choose E and M in a such a way that $E = \varepsilon[L(G)]$ and $M = \mu[L(G)]$, though of course $L(G)$ need not be identical with $E \times M$.

Example 3.1 (See also Example 2.5) 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 $f : \mathbb{N} \to L$. Define the grammar G in the same way as in Example 2.5. 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 $\mathcal{I}(s^n b) = f(n)$. Thus this grammar generates L. We conclude that a language has a grammar if and only if it is finite or countable. ⚽

We refer the reader to 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 $\mathcal{I}(f) : (E \times M)^{\Omega(f)} \hookrightarrow (E \times M)$ and so we can factor $\mathcal{I}(f)$ into a *pair* of partial functions

$$f^\varepsilon := \pi_E \circ \mathcal{I}(f), \qquad f^\mu := \pi_M \circ \mathcal{I}(f). \tag{3.16}$$

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

$$\begin{aligned} f^\varepsilon(\sigma_0, \cdots, \sigma_{\Omega(f)-1}) &:= \varepsilon(\mathcal{I}(f)(\sigma_0, \cdots, \sigma_{\Omega(f)-1})), \\ f^\mu(\sigma_0, \cdots, \sigma_{\Omega(f)-1}) &:= \mu(\mathcal{I}(f)(\sigma_0, \cdots, \sigma_{\Omega(f)-1})). \end{aligned} \tag{3.17}$$

It follows that we have

$$\mathcal{I}(f)(\sigma_0, \cdots, \sigma_{\Omega(f)-1}) = \langle f^{\varepsilon}(\sigma_0, \cdots, \sigma_{\Omega(f)-1}), f^{\mu}(\sigma_0, \cdots, \sigma_{\Omega(f)-1})\rangle. \tag{3.18}$$

This is written in a more concise form as

$$\mathcal{I}(f) = f^{\varepsilon} \times f^{\mu}. \tag{3.19}$$

Here, $f \times g$, where $f : A^n \to C$ and $g : A^n \to D$ are functions, is a function from A^n to $C \times D$ defined by

$$(f \times g)(x_0, \cdots, x_{n-1}) := \langle f(x_0, \cdots, x_{n-1}), g(x_0, \cdots, x_{n-1})\rangle. \tag{3.20}$$

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

$$\mathcal{I}^{\varepsilon}(f) := \pi_E \circ \mathcal{I}(f), \qquad \mathcal{I}^{\mu}(f) := \pi_M \circ \mathcal{I}(f). \tag{3.21}$$

As we shall see, having two independent interpretations changes things dramatically. So we shall give the new construct a name and call it a *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 $G = \langle \Omega, \mathcal{I}^{\varepsilon}, \mathcal{I}^{\mu}\rangle$ where Ω is a finite signature and $\mathcal{I}^{\varepsilon}$ and $\mathcal{I}^{\mu}$ functions that assign to a mode f two partial functions, namely $\mathcal{I}^{\varepsilon}(f) : (E \times M)^{\Omega(f)} \hookrightarrow E$ and $\mathcal{I}^{\mu}(f) : (E \times M)^{\Omega(f)} \hookrightarrow M$.

The concept of a bigrammar is a different concept, as we shall show. If $G = \langle \Omega, \mathcal{I}^{\varepsilon}, \mathcal{I}^{\mu}\rangle$ is a bigrammar then put $\mathcal{I}(f) := \mathcal{I}^{\varepsilon}(f) \times \mathcal{I}^{\mu}(f)$. Then $G_{\times} := \langle \Omega, \mathcal{I}\rangle$ is an interpreted grammar. Conversely, given an interpreted grammar $G = \langle \Omega, \mathcal{I}\rangle$, put $G^{\times} := \langle \Omega, \mathcal{I}^{\varepsilon}, \mathcal{I}^{\mu}\rangle$ as in (3.21); 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 every bigrammar H. This is because several distinct bigrammars may define the same interpreted grammar. Notice namely that

$$\mathrm{dom}(\mathcal{I}^{\varepsilon}(f) \times \mathcal{I}^{\mu}(f)) = \mathrm{dom}(\mathcal{I}^{\varepsilon}(f)) \cap \mathrm{dom}(\mathcal{I}^{\mu}(f)). \tag{3.22}$$

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

$$\mathrm{dom}(f^{\varepsilon}) = \mathrm{dom}(f^{\mu}). \tag{3.23}$$

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

$$\mathrm{dom}(\mathcal{I}^{\varepsilon}(f)) = \mathrm{dom}(\mathcal{I}^{\mu}(f)). \tag{3.24}$$

We call a bigrammar satisfying (3.24) **balanced**. The following is easy to see.

Proposition 3.1 *Let H be a bigrammar. Then H is balanced if and only if $H = (H_\times)^\times$.*

Proof Clearly, if $H = (H_\times)^\times$ then H is of the form $G^\times$ and so is balanced. Conversely, if H is balanced then $\mathrm{dom}(\mathcal{I}(f)) = \mathrm{dom}(f^\varepsilon) = \mathrm{dom}(f^\mu)$ and so we have $\mathrm{dom}(\mathcal{I}^\varepsilon(f) \times \mathcal{I}^\mu(f)) = \mathrm{dom}(\mathcal{I}(f))$. It follows that $f = \mathcal{I}^\varepsilon(f) \times \mathcal{I}^\mu(f)$ and so $H = (H_\times)^\times$. □

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 $\mathcal{I}$ what types of objects are generated. This is one of the reasons for our abstract definition of grammars using signatures. For example, given an interpreted grammar $G = \langle \Omega, \mathcal{I} \rangle$, we define the interpretation of a constant term t by induction as follows:

$$\iota_G(f s_0 s_1 \cdots s_{\Omega(f)-1}) := \mathcal{I}(f)(\iota_G(s_0), \iota_G(s_1), \cdots, \iota_G(s_{\Omega(f)-1})). \tag{3.25}$$

We use also the following notation. For terms t we let t^ε be the exponent of $\iota(t)$ and t^μ its meaning. A term t is **semantically definite** if t^μ exists; and it is **orthographically definite** if t^ε 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 indefinite, in which case t is 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 (or neither orthographically nor semantically 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 σ_i.

Example 3.2 Let $E := A^*$ where $A := \{\mathtt{0}, \mathtt{1}, +, -, (,), =\}$. Let $M := \mathbb{Z} \cup \{\top, \bot\}$. $F := \{f_0, f_1, f_2, f_3, f_4, f_5, f_6\}$. $\Omega(f_0) := \Omega(f_1) := 0$, $\Omega(f_2) := \Omega(f_3) := 1$, $\Omega(f_4) := \Omega(f_5) := \Omega(f_6) := 2$. $\vec{x}$ is **binary** if it only contains /**0**/ and /1/; $\vec{x}$ is a **term** if it does not contain /=/. The grammar is shown in Fig. 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 $\bot$). ⚽

$$
\begin{aligned}
\mathcal{J}(f_0)() &:= \langle \mathtt{0}, 0\rangle \\
\mathcal{J}(f_1)() &:= \langle 1, 1\rangle \\
\mathcal{J}(f_2)(\langle \vec{x}, m\rangle) &:= \begin{cases} \langle \vec{x}\mathtt{0}, 2m\rangle & \text{if } \vec{x} \text{ is binary,} \\ \text{undefined} & \text{else.} \end{cases} \\
\mathcal{J}(f_3)(\langle \vec{x}, m\rangle) &:= \begin{cases} \langle \vec{x}1, 2m+1\rangle & \text{if } \vec{x} \text{ is binary,} \\ \text{undefined} & \text{else.} \end{cases} \\
\mathcal{J}(f_4)(\langle \vec{x}, m\rangle, \langle \vec{y}, n\rangle) &:= \begin{cases} \langle (\vec{x}\mathtt{+}\vec{y}), m+n\rangle & \text{if } \vec{x}, \vec{y} \text{ are terms,} \\ \text{undefined} & \text{else.} \end{cases} \\
\mathcal{J}(f_5)(\langle \vec{x}, m\rangle, \langle \vec{y}, n\rangle) &:= \begin{cases} \langle (\vec{x}\mathtt{-}\vec{y}), m-n\rangle & \text{if } \vec{x}, \vec{y} \text{ are terms,} \\ \text{undefined} & \text{else.} \end{cases} \\
\mathcal{J}(f_6)(\langle \vec{x}, m\rangle, \langle \vec{y}, n\rangle) &:= \begin{cases} \langle \vec{x}\mathtt{=}\vec{y}, \top\rangle & \text{if } \vec{x}, \vec{y} \text{ are terms and } m = n, \\ \langle \vec{x}\mathtt{=}\vec{y}, \bot\rangle & \text{if } \vec{x}, \vec{y} \text{ are terms and } m \neq n, \\ \text{undefined} & \text{else.} \end{cases}
\end{aligned}
\tag{3.26}
$$

Fig. 3.1 A grammar for binary strings

Example 3.3 We shall now define an unbalanced bigrammar that defines the same interpreted language as the grammar in the previous example. The semantic functions are shown in Figs. 3.2 and 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 (3.24). For example, we find that $\mathcal{K}^\varepsilon(f_2)(\langle \mathtt{(1+1)}, 2\rangle)$ is undefined while $\mathcal{K}^\mu(f_2)(\mathtt{(1+1)}, 2\rangle) = 4$, since $\mathcal{K}^\mu$ does not look at the exponent. 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. ⚽

$$
\begin{aligned}
\mathcal{K}^\varepsilon(f_0)() &:= \mathtt{0} \\
\mathcal{K}^\varepsilon(f_1)() &:= \mathtt{1} \\
\mathcal{K}^\varepsilon(f_2)(\langle \vec{x}, m\rangle) &:= \begin{cases} \vec{x}\mathtt{0} & \text{if } \vec{x} \text{ is binary,} \\ \text{undefined} & \text{else.} \end{cases} \\
\mathcal{K}^\varepsilon(f_3)(\langle \vec{x}, m\rangle) &:= \begin{cases} \vec{x}\mathtt{1} & \text{if } \vec{x} \text{ is binary,} \\ \text{undefined} & \text{else.} \end{cases} \\
\mathcal{K}^\varepsilon(f_4)(\langle \vec{x}, m\rangle, \langle \vec{y}, n\rangle) &:= \begin{cases} (\vec{x}\mathtt{+}\vec{y}) & \text{if } \vec{x}, \vec{y} \text{ are terms,} \\ \text{undefined} & \text{else.} \end{cases} \\
\mathcal{K}^\varepsilon(f_5)(\langle \vec{x}, m\rangle, \langle \vec{y}, n\rangle) &:= \begin{cases} (\vec{x}\mathtt{-}\vec{y}) & \text{if } \vec{x}, \vec{y} \text{ are terms,} \\ \text{undefined} & \text{else.} \end{cases} \\
\mathcal{K}^\varepsilon(f_6)(\langle \vec{x}, m\rangle, \langle \vec{y}, n\rangle) &:= \begin{cases} \vec{x}\mathtt{=}\vec{y} & \text{if } \vec{x}, \vec{y} \text{ are terms,} \\ \text{undefined} & \text{else.} \end{cases}
\end{aligned}
\tag{3.27}
$$

Fig. 3.2 An unbalanced bigrammar for binary strings I

$$
\begin{aligned}
\mathcal{K}^{\mu}(f_0)() &:= 0\\
\mathcal{K}^{\mu}(f_1)() &:= 1\\
\mathcal{K}^{\mu}(f_2)(\langle \vec{x}, m\rangle) &:= \begin{cases} 2m & \text{if } m \in \mathbb{Z},\\ \text{undefined} & \text{else.}\end{cases}\\
\mathcal{K}^{\mu}(f_3)(\langle \vec{x}, m\rangle) &:= \begin{cases} 2n+1 & \text{if } m \in \mathbb{Z},\\ \text{undefined} & \text{else.}\end{cases}\\
\mathcal{K}^{\mu}(f_4)(\langle \vec{x}, m\rangle, \langle \vec{y}, n\rangle) &:= \begin{cases} m+n & \text{if } m, n \in \mathbb{Z},\\ \text{undefined} & \text{else.}\end{cases}\\
\mathcal{K}^{\mu}(f_5)(\langle \vec{x}, m\rangle, \langle \vec{y}, n\rangle) &:= \begin{cases} m-n & \text{if } m, n \in \mathbb{Z},\\ \text{undefined} & \text{else.}\end{cases}\\
\mathcal{K}^{\mu}(f_6)(\langle \vec{x}, m\rangle, \langle \vec{y}, n\rangle) &:= \begin{cases} \top & \text{if } m, n \in \mathbb{Z} \text{ and } m = n,\\ \bot & \text{if } m, n \in \mathbb{Z} \text{ and } m \neq n,\\ \text{undefined} & \text{else.}\end{cases}
\end{aligned}
\tag{3.28}
$$

Fig. 3.3 An unbalanced bigrammar for binary strings II

Let me conclude with a few words on the algebraic treatment. A grammar $G = \langle \Omega, \mathcal{I}\rangle$ can also be viewed as a partial Ω-algebra defined over the space $E \times M$ (see Appendix A for definitions). Bigrammars have no straightforward algebraic equivalent. Exercises 3.10 and 3.11 will pursue this theme.

Exercise 3.1 It is possible to interpret the modes f_2 and f_3 by the string functions $\vec{x} \mapsto 0^\frown\vec{x}$ and $\vec{x} \mapsto 1^\frown\vec{x}$. Show that it is however impossible to use the meaning functions given above with these string functions.

Exercise 3.2 (Continuing the previous exercise.) Give a grammar that generates the language of equations using the string functions above. (Evidently, the functions on meanings must be quite different.)

Exercise 3.3 Let $G = \langle \Omega, \mathcal{I}\rangle$ be a grammar. Show that there is a bigrammar $G_\bullet = \langle \Omega, \mathcal{I}^{\varepsilon}_\bullet, \mathcal{I}^{\mu}_\bullet\rangle$ such that $(G_\bullet)^\times = G$ and such that for every $f \in F$, $\mathcal{I}^{\varepsilon}_\bullet(f)$ is total. (Dually, we can construct $G_\bullet$ such that $\mathcal{I}^{\mu}_\bullet(f)$ is total for every $f \in F$.)

Exercise 3.4 (Using the previous exercise.) Show by giving an example that we cannot expect both $\mathcal{I}^{\varepsilon}_\bullet(f)$ and $\mathcal{I}^{\mu}_\bullet(f)$ 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

$$\mathcal{I}(f) = f^{\varepsilon} \times f^{\mu}, \tag{3.29}$$

with the functions defined as given in (3.17). If, however, we start with a bigrammar we simply put

$$f^{\varepsilon} := \mathcal{I}^{\varepsilon}(f), \qquad f^{\mu} := \mathcal{I}^{\mu}(f). \tag{3.30}$$

In this case, as we observed, (3.24) does not necessarily hold any more. Although we shall not mention this fact in the sequel, the reader is asked to be aware of the possibility that bigrammars can help to distribute the partiality between syntax and semantics, which is why we shall work mainly with bigrammars rather than grammars.

There are two senses in which the equation (3.29) can be required to hold. I call the first the *strict sense*: the equation is valid as stated above. This means that the equation specified is valid even if the relevant functions are applied to signs that are not in the language. The *extensional sense* requires that the equation only holds for the language of the grammar. This is formally expressed in (3.31).

$$\mathcal{I}(f) \restriction L(G) = (f^{\varepsilon} \times f^{\mu}) \restriction L(G) \tag{3.31}$$

Here, if $f : A^n \hookrightarrow B$ and $C \subseteq A$,

$$f \restriction C := \{\langle \vec{c}, f(\vec{c})\rangle : \vec{c} \in C^n\}. \tag{3.32}$$

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 in the language. 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 3.8 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 this question is studied in the so-called "wug-test", where people are asked to form the plural of a word that is not English. If a speaker forms a plural of such a word it means that his or her morphological functions are more general; they operate on words that are not English, and they operate even in the absence of any semantics. Children face 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. For example, it is conceivable that when presented with a nonexistent verbal root, German speakers will differ as to how they will inflect a new verb even when they completely agree on the inflection of existing verbs (though I am not aware of a positive result showing this).

Thus, we assume with some justification that the functions above may also be 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 $G' = \langle \Omega, \mathcal{I}' \rangle$ is an **extensional variant** of $G = \langle \Omega, \mathcal{I} \rangle$ if $L(G') = L(G)$ and for every mode f, $\mathcal{I}'(f) \restriction L(G) = \mathcal{I}(f) \restriction L(G)$. 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 us return to the equation (3.29) above. I shall rewrite it as follows:

$$\begin{aligned} &\mathcal{I}(f)(\langle e_0, m_0\rangle, \cdots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1}\rangle) \\ &= \langle f^{\varepsilon}(\langle e_0, m_0\rangle, \cdots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1}\rangle), \\ &\quad f^{\mu}(\langle e_0, m_0\rangle, \cdots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1}\rangle)\rangle. \end{aligned} \tag{3.33}$$

We say that a bigrammar is compositional if f^{μ} does not depend on the e_i. This can be restated as follows. (For notions of independence for (partial) functions see 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^{μ} is (weakly) independent of the exponents of the signs. If the f^{μ} 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 $L = L(G)$.

Example 3.4 There are pairs of words whose meaning is roughly the same, of which one member is singular and the other in the plural (see Kac, Manaster-Ramer, and Rounds (1987)): examples are /`military`/:/`armed forces`/, /`forest`/:/`woods`/ and /`location`/: /`whereabouts`/. Consider a bigrammar that has these words as values of constants and a single unary operation that forms the regular plural. Semantically, each of the concepts has a plural (there is a notion of armies, forests and locations). However, depending on the exponent, the plural can or cannot be regularly formed. This grammar is therefore semicompositional but not compositional. Using the notation of Example 2.10, the term $p(f_{\text{forest}})$ is definite and interpreted by $\langle \texttt{forests}, \mathsf{pl}'(\mathsf{forest}')\rangle$. However, $p(f_{\text{woods}})$ is not definite. It is however semantically definite. An example of a compositional bigrammar is the following. Switch the interpretation of p as follows: if the noun is in the singular, form the regular plural. If it is in the plural, leave the noun unchanged. The second grammar generates

the pluralia tanta also in their plural meaning, so that, e.g., /armed forces/ means either army (singular meaning) or armies (plural meaning). ⚽

The notion of semicompositionality may easily be confused with compositionality. The difference is not in the value that the function yields: it is unique. The difference is whether the choice of certain expressions can make the semantic function undefined when it has a value for at least *some* expressions. In a compositional bigrammar this is excluded while a semicompositional still allows for that possibility.

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 65 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

$$\mu(\mathcal{I}(f)(\sigma_0, \cdots, \sigma_{\Omega(f)-1})) \stackrel{\geq}{=} f^\mu_*(\mu(\sigma_0), \cdots, \mu(\sigma_{\Omega(f)-1})). \tag{3.34}$$

The sign $\stackrel{\geq}{=}$ means that the left- and right-hand sides are equal *if defined*; and moreover, the right-hand 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

$$f^\mu_*(m_0, \cdots, m_{\Omega(f)-1}) := f^\mu(\langle e, m_0\rangle, \langle e, m_1\rangle, \cdots, \langle e, m_{\Omega(f)-1}\rangle), \tag{3.35}$$

where e is chosen arbitrarily. Since by assumption f^μ 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

$$\mu[g] := \{\langle \mu(\sigma_0), \cdots, \mu(\sigma_n)\rangle : \langle \sigma_0, \cdots, \sigma_n\rangle \in g\}. \tag{3.36}$$

Then we may alternatively define f^μ_* by

$$f^\mu_* := \mu[\mathcal{I}(f)]. \tag{3.37}$$

Independence from the exponents guarantees that this is a function. We see here more explicitly that f^μ_* is a partial function only on meanings. Suppose now that L is compositional; this means that there is a compositional grammar G such that $L = L(G)$. This means in turn that for every $\sigma \in L$ there is a term t such that $\sigma = \iota_G(t)$. If $t = f s_0 \cdots s_{\Omega(f)-1}$ then the meaning of $\iota_G(t)$ equals $f^\mu_*(\mu(\iota_G(s_0)), \cdots, \mu(\iota_G(s_{\Omega(f)-1})))$, which is to say that, given that the σ_i are the parts of σ, the meaning of σ is the result of applying the function f^μ_* 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

$$\widehat{f^{\varepsilon}} := (\varepsilon \circ \mathcal{I}(f)) \restriction L(G). \tag{3.38}$$

This function is defined exactly on the signs of $L(G)$. Now take as $\widehat{f^{\mu}_{*}}$ any function extending f^{μ}_{*}. (In other words, $\widehat{f^{\varepsilon}}$ carries all the load in terms of undefinedness. In this case, $\widehat{f^{\mu}_{*}}$ may even be a total function.)

Example 3.5 Here is an example. Let $G = \langle \Omega, \mathcal{I} \rangle$ be a grammar containing a binary mode f and zeroary modes g_i, $i < 3$, where

$$\begin{aligned} \mathcal{I}(g_0)() &= \langle \mathtt{ed}, \mathsf{past}' \rangle \\ \mathcal{I}(g_1)() &= \langle \mathtt{laugh}, \mathsf{laugh}' \rangle \\ \mathcal{I}(g_2)() &= \langle \mathtt{car}, \mathsf{car}' \rangle \end{aligned} \tag{3.39}$$

Here, I am assuming the following type assignment: $\mathsf{car}' : e \to t$, $\mathsf{laugh}' : e \to s \to t$ and $\mathsf{past}' : (e \to s \to t) \to (e \to s \to t)$.

$$\mathcal{I}(f)(\langle e, m \rangle, \langle e', m' \rangle) := \langle e^\frown e', m'(m) \rangle \tag{3.40}$$

Note that this is undefined if $m'(m)$ is undefined. This means that semantically the only meaningful combination is $\mathsf{past}'(\mathsf{laugh}')$. Now take the bigrammar $G^{\times} = \langle \Omega, \mathcal{I}^{\varepsilon}, \mathcal{I}^{\mu} \rangle$. Define a new bigrammar $\langle \Omega, \mathcal{K}^{\varepsilon}, \mathcal{K}^{\mu} \rangle$ as follows. $\mathcal{K}^{\mu}(f)$ is any total function extending $\mathcal{I}^{\mu}(f)$; for example, it also takes the pairs $\langle e, \mathsf{car}' \rangle$, $\langle e', \mathsf{past}' \rangle$ as arguments (whatever e and e') and returns some value. Then put

$$\mathcal{K}^{\varepsilon}(f)(\langle e, m \rangle, \langle e', m' \rangle) := \begin{cases} e^\frown e' & \text{if } e = /\mathtt{laugh}/ \text{ and } e' = /\mathtt{ed}/, \\ \text{undefined} & \text{else.} \end{cases} \tag{3.41}$$

It is not hard to check that $\mathcal{K}^{\varepsilon}(f) = \mathcal{I}^{\varepsilon}(f)$. This bigrammar therefore generates the same output language. The source of partiality has been shifted from the semantics to the syntax. ⚽

A particular choice that we may take for f^{μ}_{*} is $\mu[\mathcal{I}(f)]$. 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 counts as 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^ε is weakly independent of the m_i. If f^ε are also strongly independent of the 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 $L = L(G)$.

Semiautonomy says that the exponent of a complex sign is the result of applying a certain function to the exponent of its parts and that 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

$$\varepsilon(\mathcal{I}(f)(\sigma_0, \cdots, \sigma_{\Omega(f)-1})) \stackrel{>}{=} f^\varepsilon_*(\varepsilon(\sigma_0), \cdots, \varepsilon(\sigma_{\Omega(f)-1})). \tag{3.42}$$

Again, if the left-hand side is defined then the right-hand side is as well but not conversely. In an autonomous grammar, also the converse holds.

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 are functions f^ε_* and f^μ_* such that for all $\sigma_i = \langle e_i, m_i \rangle$, $i < n$:

$$\mathcal{I}(f)(\sigma_0, \cdots, \sigma_{\Omega(f)-1}) = \langle f^\varepsilon_*(e_0, \cdots, e_{\Omega(f)-1}), f^\mu_*(m_0, \cdots, m_{\Omega(f)-1}) \rangle. \tag{3.43}$$

with the left-hand side defined if and only if the right-hand side is. (The functions f^ε_* and f^μ_* are defined as $\varepsilon[\mathcal{I}^\varepsilon(f)]$ and $\mu[\mathcal{I}^\mu(f)]$, respectively.) Another formulation is

$$\mathcal{I}(f) = (f^\varepsilon_* \circ \langle \overbrace{\varepsilon, \cdots, \varepsilon}^{\Omega(f)} \rangle) \times (f^\mu_* \circ \langle \overbrace{\mu, \cdots, \mu}^{\Omega(f)} \rangle) \tag{3.44}$$

or

$$\mathcal{I}(f)(\sigma_0, \cdots, \sigma_{\Omega(f)-1}) \tag{3.45}$$
$$= \langle f_*^{\varepsilon}(\varepsilon(\sigma_0), \cdots, \varepsilon(\sigma_{\Omega(f)-1})),\ f_*^{\mu}(\mu(\sigma_0), \cdots, \mu(\sigma_{\Omega(f)-1}))\rangle.$$

It may be thought that for languages, extensional independence follows from extensional autonomy and extensional compositionality. However, this does not seem to be the case. I remark here that I have not been able to find an example of a language that is *not* (!) independent. If there are no restrictions on the functions that can be used, independence seems to be guaranteed.

Example 3.6 We construct various different grammars to show that autonomy and compositionality are independent notions. Let $A := \{\mathsf{a}\}$, $E := A^*$; $M := \mathbb{N}$. The signature is $\{f_0, f_1, f_2\}$, with f_0 zeroary and f_1 and f_2 both unary. We have

$$\begin{aligned}
\mathcal{I}(f_0)() &:= \langle \varepsilon, 0\rangle \\
\mathcal{I}(f_1)(\langle \vec{x}, n\rangle) &:= \begin{cases} \langle \vec{x}{}^\frown\mathsf{a}, n+1\rangle & \text{if } |\vec{x}| = n, \\ \text{undefined} & \text{otherwise.} \end{cases} \\
\mathcal{I}(f_2)(\langle \vec{x}, n\rangle) &:= \begin{cases} \langle \vec{x}{}^\frown\mathsf{a}, n\rangle & \text{if } |\vec{x}| \geq n, \\ \langle \vec{x}, n+1\rangle & \text{otherwise.} \end{cases}
\end{aligned} \tag{3.46}$$

Call this grammar U. The action of the unary functions on the space $E \times M$ is shown in Fig. 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 $\mathcal{I}(f_2)$ is never used inside D. Thus, we could have made $\mathcal{I}(f_2)(\langle \vec{x}, n\rangle)$ undefined if $n > |\vec{x}|$. This would give us an extensional variant of the original grammar. U is not autonomous: $\mathcal{I}(f_2)(\langle \mathsf{a}, 3\rangle) = \langle \mathsf{a}, 4\rangle$ but $\mathcal{I}(f_2)(\langle \mathsf{a}, 1\rangle) = \langle \mathsf{aa}, 1\rangle$. So to compute the exponent we need to know the meaning. It is not compositional either. For we have in addition to $\mathcal{I}(f_2)(\langle \mathsf{a}, 3\rangle) = \langle \mathsf{a}, 4\rangle$ also $\mathcal{I}(f_2)(\langle \mathsf{aaa}, 3\rangle) = \langle \mathsf{aaaa}, 3\rangle$, so to compute the meaning we need to know the exponent.

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

$$\begin{aligned}
\mathcal{I}^a(f_2)(\langle \vec{x}, n\rangle) &:= \begin{cases} \langle \vec{x}{}^\frown\mathsf{a}, n\rangle & \text{if } |\vec{x}| \geq n, \\ \langle \vec{x}{}^\frown\mathsf{a}, n+1\rangle & \text{else.} \end{cases} \\
\mathcal{I}^c(f_2)(\langle \vec{x}, n\rangle) &:= \begin{cases} \langle \vec{x}{}^\frown\mathsf{a}, n\rangle & \text{if } |\vec{x}| \geq n, \\ \langle \vec{x}{}^\frown\mathsf{a}{}^\frown\mathsf{a}, n\rangle & \text{else.} \end{cases} \\
\mathcal{I}^{ac}(f_2)(\langle \vec{x}, n\rangle) &:= \langle \vec{x}{}^\frown\mathsf{a}, n\rangle
\end{aligned} \tag{3.47}$$

All of them only generate the language D. The grammar $U^{ac} := \langle \Omega, \mathcal{I}^{ac}\rangle$ is semiautonomous and semicompositional.

$U^c = \langle \Omega, \mathcal{I}^c\rangle$ is semicompositional but not semiautonomous. To see this, note that we have $\mu(\mathcal{I}^c(f_2)(\langle e, m\rangle)) = m$, which is independent of e; on the other hand

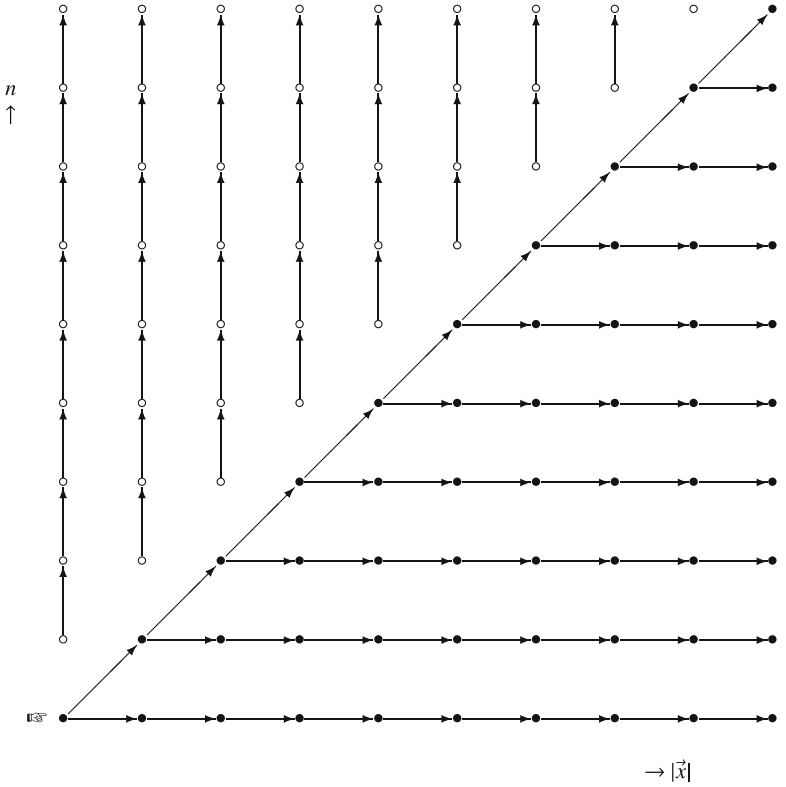

Fig. 3.4 The action of the grammar U

we have $\varepsilon(\mathcal{I}^c(f_2)(\langle \mathtt{aa}, 2\rangle)) = \mathtt{aaa} \neq \mathtt{aa} = \varepsilon(\mathcal{I}^c(f_2)(\langle \mathtt{aa}, 3\rangle))$. Similarly we find that $\langle U^a := \langle \Omega, \mathcal{I}^a\rangle$ is semiautonomous but not semicompositional. ⚽

Now, let $\mathcal{J}(f_0) := \mathcal{I}(f_0)$ and $\mathcal{J}(f_2) := \mathcal{I}(f_2)$. Put

$$\mathcal{J}(f_1)(\langle \vec{x}, n\rangle) := \langle \vec{x}^\frown \mathtt{a}, n+1\rangle. \tag{3.48}$$

Define $\mathcal{J}^a$, $\mathcal{J}^c$ and $\mathcal{J}^{ac}$ by changing the interpretation of f_2 as above. $\langle \Omega, \mathcal{J}^{ac}\rangle$ is independent, that is, autonomous and compositional. Similarly, $\mathcal{J}^a$ is autonomous and noncompositional while $\mathcal{J}^c$ is nonautonomous but compositional.

Finally, let us look at these concepts for bigrammars. If a bigrammar is autonomous then it is possible to define an extensional variant of the form $\langle \Omega, \mathcal{I}^\varepsilon_\circ, \mathcal{I}^\mu_\circ\rangle$ where $\mathcal{I}^\varepsilon_\circ(f)$ is total for every f. Namely, observe that there is a function g on exponents such that

$$\mathcal{I}^\varepsilon(f)(\vec{\sigma}) = g(e_0, \cdots, e_{\Omega(f)-1}). \tag{3.49}$$

Choose a total extension $g_\circ \supseteq g$.

$$\begin{aligned} \mathcal{I}^{\varepsilon}_{\circ}(f)(\vec{\sigma}) &:= g_{\circ}(e_0, \cdots, e_{\Omega(f)-1}) \\ \mathcal{I}^{\mu}_{\circ}(f) &:= \mathcal{I}^{\mu}(f) \restriction \mathrm{dom}(\mathcal{I}^{\varepsilon}(f)) \end{aligned} \tag{3.50}$$

Then $\mathcal{I}^{\varepsilon}_{\circ}(f)(\vec{\sigma})$ is defined if and only if $\vec{\sigma} \in \mathrm{dom}(\mathcal{I}^{\mu}_{\circ}(f)) = \mathrm{dom}(\mathcal{I}^{\varepsilon}(f)) \cap \mathrm{dom}(\mathcal{I}^{\mu}(f))$. And in this case

$$\begin{aligned} \langle \mathcal{I}^{\varepsilon}_{\circ}(f)(\vec{\sigma}), \mathcal{I}^{\mu}_{\circ}(f)(\vec{\sigma}) \rangle &= \langle g_{\circ}(\vec{e}), \mathcal{I}^{\mu}(f)(\vec{\sigma}) \rangle \\ &= \langle g(\vec{e}), \mathcal{I}^{\mu}(f)(\vec{\sigma}) \rangle \\ &= \langle \mathcal{I}^{\varepsilon}(f)(\vec{\sigma}), \mathcal{I}^{\mu}(f)(\vec{\sigma}) \rangle \end{aligned} \tag{3.51}$$

Example 3.7 From a grammar we can construct two bigrammars where the partiality is only in one component: 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 $A := \{\mathsf{a}\}$, $E := A^*$; $M := \mathbb{N}$. The signature is $\{f_0, f_1, f_2\}$, with f_0 zeroary and f_1 and f_2 both unary.

$$\begin{aligned} \mathcal{I}(f_0)() &:= \langle \varepsilon, 0 \rangle \\ \mathcal{I}(f_1)(\langle \vec{x}, n \rangle) &:= \langle \vec{x}^\frown \mathsf{a}, n+1 \rangle \\ \mathcal{I}(f_2)(\langle \vec{x}, n \rangle) &:= \begin{cases} \langle \vec{x}^\frown \mathsf{a}, n \rangle & \text{if } |\vec{x}| = n, \\ \text{undefined} & \text{else.} \end{cases} \end{aligned} \tag{3.52}$$

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

$$\begin{aligned} \mathcal{I}^{\ltimes}_{\varepsilon}(f_1)(\langle \vec{x}, n \rangle) &:= \vec{x}^\frown \mathsf{a} \\ \mathcal{I}^{\ltimes}_{\varepsilon}(f_2)(\langle \vec{x}, n \rangle) &:= \begin{cases} \vec{x}^\frown \mathsf{a} & \text{if } |\vec{x}| = n, \\ \text{undefined} & \text{else.} \end{cases} \end{aligned} \tag{3.53}$$

$$\begin{aligned} \mathcal{I}^{\ltimes}_{\mu}(f_1)(\langle \vec{x}, n \rangle) &:= n+1 \\ \mathcal{I}^{\ltimes}_{\mu}(f_2)(\langle \vec{x}, n \rangle) &:= n \end{aligned} \tag{3.54}$$

The second bigrammar is as follows.

$$\begin{aligned} \mathcal{I}^{\rtimes}_{\varepsilon}(f_1)(\langle \vec{x}, n \rangle) &:= \vec{x}^\frown \mathsf{a} \\ \mathcal{I}^{\rtimes}_{\varepsilon}(f_2)(\langle \vec{x}, n \rangle) &:= \vec{x}^\frown \mathsf{a} \end{aligned} \tag{3.55}$$

$$\begin{aligned} \mathcal{I}^{\rtimes}_{\mu}(f_1)(\langle \vec{x}, n \rangle) &:= n+1 \\ \mathcal{I}^{\rtimes}_{\mu}(f_2)(\langle \vec{x}, n \rangle) &:= \begin{cases} n & \text{if } |\vec{x}| = n, \\ \text{undefined} & \text{else.} \end{cases} \end{aligned} \tag{3.56}$$

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

$$\mathcal{I}_{\varepsilon}^{\ltimes}(f_2)(\langle \mathtt{aaa}, 2\rangle) \text{ is undefined}, \qquad \mathcal{I}_{\varepsilon}^{\ltimes}(f_2)(\langle \mathtt{aaa}, 3\rangle) = \mathtt{aaaa}. \tag{3.57}$$

Thus $G^{\ltimes}$ 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 $G^{\bowtie} = \langle \Omega, \mathcal{I}_{\varepsilon}^{\bowtie}, \mathcal{I}_{\mu}^{\bowtie}\rangle$ is such a grammar. Then for any given string $\vec{x}$ there is some n (namely $|\vec{x}|$) such that $\mathcal{I}_{\varepsilon}^{\bowtie}(f_2)(\langle \vec{x}, n\rangle)$ is defined. If the grammar is autonomous this means that for all m $\mathcal{I}_{\varepsilon}^{\bowtie}(f_2)(\langle \vec{x}, m\rangle)$ is defined. Hence the function $\mathcal{I}_{\varepsilon}^{\bowtie}(f_2)$ is total. Likewise we see that $\mathcal{I}_{\mu}^{\bowtie}(f_2)$ is total. It follows that $\mathrm{dom}(\mathcal{I}^{\bowtie}(f_2)) = \mathrm{dom}(\mathcal{I}(f_2))$ equals $E \times M$. But this is not the case in G. ⚽

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 3.8 This and the next example deal with the problem of reduplication. In Malay, the plural of a noun is formed by reduplication: /orang/ means "man", /orang-orang/ means "men" (see also the discussion on page 52). Thus, the plural mode p in Malay is a unary mode and is interpreted as follows.

$$\mathcal{I}(p)(\langle e, m\rangle) := \begin{cases} \langle e^{\frown}\text{-}^{\frown}e, \mathsf{pl}'(m)\rangle & \text{if } e \text{ is a singular noun,} \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{3.58}$$

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.

$$\mathcal{I}(q)(\langle e, m\rangle, \langle e', m'\rangle) := \begin{cases} \langle e^{\frown}\text{-}^{\frown}e', \mathsf{pl}'(m)\rangle & \text{if } e \text{ and } e' \text{ are nouns} \\ & \quad \text{and } e = e', \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{3.59}$$

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

$$\mathcal{I}(r)(\langle e, m\rangle, \langle e', m'\rangle) := \begin{cases} \langle e^{\frown}\text{-}^{\frown}e', \mathsf{pl}'(m)\rangle & \text{if } e \text{ and } e' \text{ are nouns} \\ & \quad \text{and } m = m', \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{3.60}$$

This looks similar to q but the difference is that the combinatorial restrictions are now semantic and syntactic rather than only 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 then our theory is 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, q or r (though it is in principle also possible to treat exceptions with extra modes as well). ⚽

The previous discussion uses grammars but it is clear how the bigrammars in question should be constructed.

Example 3.9 Manaster-Ramer (1986) discuss a construction of English in which a constituent is repeated verbatim:

`The North Koreans were developing nuclear weapons anyway, Iraq war or no Iraq war.` (3.61)

*`The North Koreans were developing nuclear weapons anyway, Iraq war or no Afghanistan war.` (3.62)

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.

$$\mathcal{I}(f)(\langle e, m\rangle) := \begin{cases} \langle e^\frown\!\texttt{␣or␣no␣}^\frown e, \mathsf{irrespective\text{-}of}'(m)\rangle & \text{if } e \text{ is an NP,} \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{3.63}$$

I put aside the alternative with a binary operation that checks for string identity. This construction 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 $/\vec{x}$ `or no` $\vec{y}/$, where $\vec{x}$ and $\vec{y}$ may be different but must be synonymous. I shall leave the issue of nonidentity aside and focus on the following point. What Pullum and Rawlins (2007) propose is that rather than checking syntactic identity, English works with a binary mode g defined by

$$\mathcal{I}(f)(\langle e, m\rangle, \langle e', m'\rangle) := \begin{cases} \langle e^\frown \texttt{␣or␣no␣}^\frown e, & \text{if } e, e' \text{ are NP} \\ \quad \textsf{irrespective-of}'(m)\rangle & \text{and } m = m', \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{3.64}$$

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 $\langle \Omega, \mathcal{I}^\varepsilon, \mathcal{I}^\mu\rangle$ we may either look at the language generated by $\langle \Omega, \mathcal{I}^\varepsilon_*\rangle$ (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 empirically. 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

$$\texttt{Abelian group or no commutative group} \tag{3.65}$$

cannot mean "irrespective of an abelian group", whereas if semantic identity counted, this would be perfect. I have not investigated this, though. ⚽

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 CF string language has the following property:

Lemma 3.1 (Pumping Lemma) *Let L be a context free string language. 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 others (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 $\langle \Omega, \mathcal{I}^\varepsilon, \mathcal{I}^\mu\rangle$. Recall from Definition 2.23 the definition of a categorial equivalence. The first is the equivalence $\sim_{G^\varepsilon}$, where $G^\varepsilon := \langle G, \mathcal{I}^\varepsilon \times 1\rangle$, 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 2.23, 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, $\mathcal{I}(f)(\vec{\sigma})$ is defined if only $\mathcal{I}^\varepsilon(f)(\vec{e})$ is defined.

Theorem 3.1 *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 of 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

$$\iota_G(r(s^n(t))) = \langle \vec{u}\vec{y}^n\vec{v}\vec{z}^n\vec{w}, f(g^n(p))\rangle \in L. \tag{3.66}$$

Example 3.10 The assumption of the syntactic well regimentedness cannot be dropped. Here is an example. Let $E := \mathsf{v}^*$. According to Thue (1914) there is an infinite word $w_0w_1w_2\cdots$ over $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ such that no finite subword is immediately repeated. Let $M := \{w_0w_1\cdots w_{n-1} : n \in \mathbb{N}\}$. Our language is $\{\langle \mathsf{v}^n, w_0w_1\cdots w_{n-1}\rangle : n \in \mathbb{N}\}$. 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:

$$\begin{aligned}
&\mathcal{I}^\varepsilon_*(p)() := \varepsilon \qquad \mathcal{I}^\mu(p)() := \varepsilon\\
&\mathcal{I}^\varepsilon_*(f_a)(\vec{x}) := \vec{x}^\frown\mathsf{v}\ \mathcal{I}^\mu_*(f_a)(\vec{x}) := \begin{cases}\vec{x}^\frown\mathsf{a} & \text{if } \vec{x}^\frown\mathsf{a} \in M,\\ \text{undefined} & \text{else.}\end{cases}\\
&\mathcal{I}^\varepsilon_*(f_b)(\vec{x}) := \vec{x}^\frown\mathsf{v}\ \mathcal{I}^\mu_*(f_b)(\vec{x}) := \begin{cases}\vec{x}^\frown\mathsf{b} & \text{if } \vec{x}^\frown\mathsf{b} \in M,\\ \text{undefined} & \text{else.}\end{cases}\\
&\mathcal{I}^\varepsilon_*(f_c)(\vec{x}) := \vec{x}^\frown\mathsf{v}\ \mathcal{I}^\mu_*(f_c)(\vec{x}) := \begin{cases}\vec{x}^\frown\mathsf{c} & \text{if } \vec{x}^\frown\mathsf{c} \in M,\\ \text{undefined} & \text{else.}\end{cases}
\end{aligned} \tag{3.67}$$

Suppose that the assertion of Theorem 3.1 holds for L. Then with the notation as in the theorem we would have

$$\sigma := \langle \vec{u}\,\vec{y}^2\,\vec{v}\,\vec{z}^2\,\vec{w}, f(g^2(p))\rangle \in L. \tag{3.68}$$

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 $\sigma \notin L$, since no string can repeat itself in a string from M. ⚽

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 Theorem 3.1 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 60) then Theorem 3.1 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 generalized to interpreted languages in the same way (Ogden's Lemma, the strengthened form of Manaster-Ramer, Moshier, and Zeitman (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 3.5 Show how to generate the language of Example 3.7 using an independent grammar.

Exercise 3.6 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 3.7 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 is already extensionally compositional.

Exercise 3.8 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 3.9 Let $L \subseteq E \times M$ be a language such that $\varepsilon[L]$ is finite. Show that L is independent. (Similarly, show that L is independent if $\mu[L]$ is finite.)

Exercise 3.10 The following exercise points at some algebraic connections. I refer to Appendix A for basic algebraic concepts. Let E and M be given. Given a signature Ω, we can think of a grammar as a partial Ω-algebra $\mathfrak{G} = \langle E \times M, I\rangle$. Now show the following. (a) G is autonomous if and only if the map ε is a homomorphism from $\mathfrak{G}$ onto some algebra $\mathfrak{E} = \langle E, J\rangle$ of exponents; can you identify the functions $J(f)$? (b) G is compositional if and only if μ is a homomorphism from $\mathfrak{G}$ onto some algebra $\langle M, K\rangle$ of meanings. Can you identify $K(f)$? *Hint.* (b) is dual to (a).

Exercise 3.11 (Continuing the previous exercise.) 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 σ. This is in line with Keenan and Stabler (2001), Pollard and Sag (1994), Mel'čuk (1993–2000), not to mention Categorial 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 relates form with meaning. Whatever this says in practice for 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 that are the result of partiality in the operations (see Kracht (2006)). So, we can in principle do without them but they make 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 them later. One result is that in case the set of signs contains only finitely many categories they can be eliminated (Theorem 3.2), though we may be forced to pay a price.

The formal details are as follows. A **c-sign** is a triple $\gamma = \langle e, c, m\rangle$. The space of c-signs is a product $E \times C \times M$. The projections will be denoted as follows.

$$\varepsilon(\langle e, c, m\rangle) := e, \qquad \kappa(\langle e, c, m\rangle) := c, \qquad \mu(\langle e, c, m\rangle) := m. \tag{3.69}$$

Put $H := \varepsilon \times \mu$, that is,

$$H(\gamma) := \langle e, m\rangle. \tag{3.70}$$

A **c-language** is a set of c-signs. 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 $(E \times C \times M)^{\Omega(f)} \hookrightarrow (E \times C \times M)$. As before, the concept we shall be working with is slightly different.

Definition 3.8 A **trigrammar** over $E \times C \times M$ is a quadruple $\langle \Omega, \mathcal{I}^{\varepsilon}, \mathcal{I}^{\kappa}, \mathcal{I}^{\mu}\rangle$, where Ω is a signature and $\mathcal{I}^{\varepsilon}(f) : (E \times C \times M)^{\Omega(f)} \to E$ an interpretation

of f in E, $\mathcal{I}^\kappa(f) : (E \times C \times M)^{\Omega(f)} \to C$ an interpretation of f in C and $\mathcal{I}^\mu(f) : (E \times C \times M)^{\Omega(f)} \to M$ an interpretation of f in M.

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

$$G_\times := \langle \Omega, \mathcal{I}^\varepsilon \times \mathcal{I}^\kappa \times \mathcal{I}^\mu \rangle. \tag{3.71}$$

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.

A trigrammar is **autonomous** if the exponent of $\mathcal{I}(f)(\vec{\sigma})$ is strongly independent of the categories and meanings of the input signs; it is **compositional** if the meaning of $\mathcal{I}(f)(\vec{\sigma})$ 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 $\mathcal{I}(f)(\vec{\sigma})$ is strongly independent of the exponents and the meanings of the input signs. The trigrammar is **independent** if it is autonomous, compositional and categorially autonomous. In case of independence we can exchange the functions f^ε, f^κ, f^μ by their reductions $f^\varepsilon_* : E^{\Omega(f)} \to E$, $f^\kappa_* : C^{\Omega(f)} \to C$, $f^\mu_* : M^{\Omega(f)} \to M$, which are obtained by removing the other components.

Let $L = L(G)$ for some trigrammar G. The H-image of L is

$$\begin{aligned} H[L] &:= \{H(\gamma) : \gamma \in L\} \\ &= \{\langle e, m\rangle : \text{there is } c \in C : \langle e, c, m\rangle \in L\}. \end{aligned} \tag{3.72}$$

The question is whether there is an interpreted grammar for $H[L]$.

Theorem 3.2 *Let $G = \langle \Omega, \mathcal{C}\rangle$ 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

$$F^+_n := \{f_{\vec{c}} : f \in F_n, \vec{c} \in C^n\}. \tag{3.73}$$

For example

$$\begin{aligned} F^+_0 &= \{f_{\langle\rangle} : f \in F_0\}, \\ F^+_1 &:= \{f_{\langle c\rangle} : f \in F_1, c \in C\}, \\ F^+_2 &:= \{f_{\langle c,c'\rangle} : f \in F_2, c, c' \in C\}. \end{aligned} \tag{3.74}$$

As for the signature, we put

$$\Omega^+(f_{\vec{c}}) := \Omega(f). \tag{3.75}$$

We define the actions of the functions over this signature.

$$\begin{aligned}\mathcal{I}(f_{c_0,c_1,\cdots,c_{n-1}})(\langle e_0, m_0\rangle, \langle e_1, m_1\rangle, \cdots, \langle e_{n-1}, m_{n-1}\rangle)\\ := H(\mathcal{C}(f)(\langle e_0, c_0, m_0\rangle, \langle e_1, c_1, m_1\rangle, \cdots, \langle e_{n-1}, c_{n-1}, m_{n-1}\rangle))\end{aligned} \tag{3.76}$$

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

$$\mathcal{I}(f_{\vec{c}})(H(\sigma_0), H(\sigma_1), \cdots, H(\sigma_{n-1})) := H(\mathcal{C}(f)(\sigma_0, \sigma_1, \cdots, \sigma_{n-1})). \tag{3.77}$$

Here the left-hand side is defined if and only if the right-hand side is; and in this case the left-hand side is defined to be whatever the right-hand side is. This defines the grammar $K := \langle \Omega, \mathcal{I}\rangle$.

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 Ω 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 a constant. Then $f^+ := f_{\langle\rangle}$. Now, $\iota_K(f^+) = H(\iota_G(f))$, by construction. Inductive case. $t = fs_0s_1\cdots s_{n-1}$. $\Omega(f) = n > 0$. Let $\iota_G(s_i) = \langle e_i, c_i, m_i\rangle$. By induction hypothesis, for every $i < n$ there is a term s_i^+ such that $\iota_K\left(s_i^+\right) = H(\iota_G(s_i))$. Then $\mathcal{C}(f)$ is defined on the $\iota_G(s_i)$ and therefore $\mathcal{I}(f_{c_0,c_1,\cdots,c_{n-1}})$ is defined on $\langle e_i, m_i\rangle = \iota_K\left(s_i^+\right)$ and yields the value

$$\begin{aligned}\iota_K(t^+) &= \mathcal{I}(f_{\vec{c}})\left(\iota_K\left(s_0^+\right), \iota_K\left(s_1^+\right), \cdots, \iota_K\left(s_{n-1}^+\right)\right)\\ &= \mathcal{I}(f_{\vec{c}})(\langle e_0, m_0\rangle, \cdots, \langle e_{n-1}, m_{n-1}\rangle))\\ &= H(\mathcal{C}(f)(\langle e_0, c_0, m_0\rangle, \cdots, \langle e_{n-1}, c_{n-1}, m_{n-1}\rangle))\\ &= H(\mathcal{C}(f)(\iota_G(s_0), \iota_G(s_1), \cdots, \iota_G(s_{n-1})))\\ &= H(\iota_G(t))\\ &= H(\sigma)\end{aligned} \tag{3.78}$$

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:

$$(f_{\vec{c}}s_0\cdots s_{\Omega(f)-1})^- := fs_0^-s_1^-\cdots s_{\Omega(f)-1}^-. \tag{3.79}$$

In particular, $(f_{\langle\rangle})^- = f$. 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 $\iota_G\left(s_i^-\right) = c_i$ and $\vec{c} = \langle c_0, c_1, \cdots, c_{n-1}\rangle$. Then, using (3.77):

$$\begin{aligned} H(\iota_G(t^-)) &= H\big(\iota_G\big(f s_0^- \cdots s_{n-1}^-\big)\big) \\ &= H\big(\mathcal{C}(f)\big(\iota_G\big(s_0^-\big), \cdots, \iota_G\big(s_{n-1}^-\big)\big)\big) \\ &= \mathcal{I}(f_{\vec{c}})\big(H\big(\iota_G\big(s_0^-\big)\big), H\big(\iota_G\big(s_1^-\big)\big), \cdots, H\big(\iota_G\big(s_{n-1}^-\big)\big)\big) \\ &= \mathcal{I}(f_{\vec{c}})(\iota_K(s_0), \iota_K(s_1), \cdots, \iota_K(s_{n-1})) \\ &= \iota_K(t) \end{aligned} \tag{3.80}$$

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. Categorial 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: $\langle$leak, n, $m\rangle$ and $\langle$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 $\langle$leak, $m\rangle$ and $\langle$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 $\langle$leak, v, $\hat{m}\rangle$ but not to the sign $\langle$leak, n, $m\rangle$. It is the interpretation of some mode

f. This mode is now replaced among others by a mode f_v, which takes as input only the sign $\langle \texttt{leak}, \hat{m}\rangle$ and forms the sign $\langle \texttt{leaked}, \mathsf{past}'(\hat{m})\rangle$. It is not defined on $\langle \texttt{leak}, m\rangle$. 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 Categorial Grammar, for example, which employs something that may be dubbed categorial well-regimentation; it means that the categories alone can tell whether a term is definite. 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 repeat Definition 3.5 almost verbatim. The following example now shows that compositionality and autonomy can be lost under reduction.

Example 3.11 Our example is based on the grammar of Example 3.7. We introduce a set $C = \{o, p\}$ of categories. For any given triple $\langle e, c, m\rangle$ we define

$$\begin{aligned} \mathcal{K}(f_1)(\langle e, c, m\rangle) &:= \begin{cases} \langle e^\frown \mathtt{a}, p, m+1\rangle & \text{if } c = p, \\ \text{undefined} & \text{else.} \end{cases} \\ \mathcal{K}(f_2)(\langle e, c, m\rangle) &:= \begin{cases} \langle e^\frown \mathtt{a}, o, m\rangle & \text{if } c = p, \\ \text{undefined} & \text{else.} \end{cases} \end{aligned} \tag{3.81}$$

From this grammar we can define the following independent trigrammar. Let $(f_i)^{\varepsilon}_{*}(e) := e^\frown \mathtt{a}$, $(f_1)^{\mu}_{*}(m) := m + 1$, $(f_2)^{\mu}_{*}(m) := m$ and, finally, $(f_1)^{\kappa} : p \mapsto p, o \mapsto\downarrow$ (= undefined), $(f_2)^{\kappa} : p \mapsto o, o \mapsto\downarrow$. Call this trigrammar K. K is independent, its reduction via H is not; it also is neither autonomous (only extensionally autonomous) nor compositional (only extensionally compositional). For the reduction is exactly the grammar of Example 3.7. ⚽

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

$p : E \to \wp(C)$ and $q : M \to \wp(C)$ 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 $p(e) \cap q(m) = \varnothing$. 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 3.2 *Let L be an independent c-language that allows to guess categories. Suppose further that L has only finitely many categories. Then $H[L]$ is independent.*

Proof Let $p : E \to \wp(C)$ and $q : M \to \wp(C)$ be the guessing functions. Let G be an independent c-grammar for L. By assumption, for every mode f there are three functions f^{ε}_{*}, f^{κ}_{*} and f^{μ}_{*} such that

$$\begin{aligned}\mathcal{I}(f)(\langle e_0, c_0, m_0\rangle, \cdots, \langle e_{n-1}, c_{n-1}, m_{n-1}\rangle) \\ = \langle f^{\varepsilon}_{*}(e_0, \cdots, e_{n-1}),\, f^{\kappa}_{*}(c_0, \cdots, c_{n-1}),\, f^{\mu}_{*}(m_0, \cdots, m_{n-1})\rangle.\end{aligned} \tag{3.82}$$

Proceed as in the proof of Theorem 3.2. 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 $\mathcal{I}(f)() = \langle e, c, m\rangle$ let $\mathcal{I}(f_{\langle\rangle})() := \langle e, m\rangle$. Now suppose that $n > 0$. For each n-ary sequence of elements from C we introduce a new mode $f_{\vec{c}}$. We set

$$(f_{\vec{c}})^{\varepsilon}(e_0, \cdots, e_{n-1}) := \begin{cases} f^{\varepsilon}_{*}(e_0, \cdots, e_{n-1}) & \text{if for every } i < n\text{: } c_i \in p(e_i) \\ & \text{and } f^{\kappa}_{*}(\vec{c}) \text{ is defined,} \\ \text{undefined} & \text{else.} \end{cases} \tag{3.83}$$

Likewise we put

$$(f_{\vec{c}})^{\mu}_{*}(m_0, \cdots, m_{n-1}) := \begin{cases} f^{\mu}_{*}(m_0, \cdots, m_{n-1}) & \text{if for every } i < n\text{: } c_i \in q(m_i) \\ & \text{and } f^{\kappa}_{*}(\vec{c}) \text{ is defined,} \\ \text{undefined} & \text{else.} \end{cases} \tag{3.84}$$

This defines the grammar G^{+} over the signature Ω^{+}. 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 claim (b) is certainly true. Now let $t = f(u_0, \cdots, u_{n-1})$, 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 $p(e_j) \cap q(m_j) = \varnothing$. Then for every sequence $\vec{c}$ either $f^{\varepsilon}_{\vec{c}}(e_0, \cdots, e_{n-1})$ or $f^{\mu}_{\vec{c}}(m_0, \cdots, m_{n-1})$ is undefined. Hence none of the functions

$\mathcal{I}(f_{\vec{c}})$ 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 $\vec{g} := \langle g_0, \cdots, g_{n-1}\rangle$ both $f_{\vec{g}}^{\varepsilon}(e_0, \cdots, e_{n-1})$ and $f_{\vec{g}}^{\mu}(m_0, \cdots, m_{n-1})$ are defined and they equal $f_*^{\varepsilon}(e_0, \cdots, e_{n-1})$ and $f_*^{\mu}(m_0, \cdots, m_{n-1})$, respectively. Since $f_*^{\kappa}(g_0, \cdots, g_{n-1})$ is also defined (by definition of the functions $f_{\vec{g}}^{\varepsilon}$ and $f_{\vec{g}}^{\mu}$) the following value exists:

$$\langle f_*^{\varepsilon}(e_0, \cdots, e_{n-1}), f_*^{\kappa}(g_0, \cdots, g_{n-1}), f_*^{\mu}(m_0, \cdots, m_{n-1})\rangle. \tag{3.85}$$

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

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. Categorial 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 λ-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 complex 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 language; 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^ε, which consists just in the functions f^ε for $f \in F$. Since f^ε may depend on the meanings of the input signs, this makes immediate sense only for an autonomous bigrammar. Recall that for such a grammar the functions f^ε_* are defined on $E^{\Omega(f)}$ with values in E. Even in this case, however, it may happen that $L(G^\varepsilon) \neq \varepsilon[L]$ precisely because there might be terms that are orthographically but not semantically definite. (In general, only $\varepsilon[L] \subseteq L(G^\varepsilon)$ holds.)

Recall from previous discussions that in grammars the domains of f^μ and f^ε are identical. In this case some of the distinctions that are of interest in this section cannot be made, such as the distinction between weak dependency of f^ε on exponents and the weak dependency of f^μ 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 defined context freeness of a string grammar intrinsically. The results in this section use the term the "context free" in this sense. The results are often more general, applying to concatenative grammars as well. I occasionally point out where results can be generalized.

Definition 3.9 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 3.12 An example of an interpreted language that is weakly but not autonomously CF. Let

$$L := \left\{ \langle \mathsf{a}^n, i \rangle : n \in \mathbb{N}, i < 2^{2^n} \right\}. \tag{3.86}$$

Given a string $\vec{x}$ of length n the number of terms that unfold to $\vec{x}$ is at most 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 $2^{2^n} > 2^{pn}$. ⚽

Theorem 3.3 *Let L be unambiguous. Then if L is weakly $\mathcal{C}$ it is also autonomously $\mathcal{C}$.*

Proof By assumption, there is a function $b : E \to M$ such that $\langle e, m\rangle \in L$ iff $m = b(e)$ (in set theory, L *is* that function b). Also, by assumption there is a string grammar $G = \langle \Omega, \mathcal{I}\rangle$ for $\varepsilon[L]$, which is in $\mathcal{C}$. Now put

$$\begin{aligned}\mathcal{I}^{\varepsilon}(f)(\langle e_0, m_0\rangle, \cdots, \langle e_{\Omega(f)n-1}, m_{\Omega(f)-1}\rangle) &:= \mathcal{I}(f)(e_0, \cdots, e_{\Omega(f)-1})\\ \mathcal{I}^{\mu}(f)(\langle e_0, m_0\rangle, \cdots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1}\rangle) &:= b(\mathcal{I}(f)(e_0, \cdots, e_{\Omega(f)-1}))\end{aligned} \tag{3.87}$$

The bigrammar $G^+ := \langle \Omega, \mathcal{I}^{\varepsilon}, \mathcal{I}^{\mu}\rangle$ is obviously strongly autonomous. Moreover, it generates L. By construction, if it generates $\langle e, m\rangle$ then (1) $e \in L(G) = E$ and (2) $m = b(e)$. Moreover, if $\langle e, m\rangle \in L$ then $m = b(e)$ and $e \in L(G)$. It follows that $\langle e, m\rangle \in L(G^+)$. □

We can strengthen this as follows.

Theorem 3.4 *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 f^{ε}_* is independent of the meanings. The art is in defining the semantic functions. By assumption, choosing $E := \varepsilon[L]$ and $M := \mu[L]$, there is a bijection $\pi : E \to M$ such that $L = \{\langle e, \pi(e)\rangle : e \in \varepsilon[L]\}$. With the help of this bijection put

$$f^{\mu}_*(m_0, \cdots, m_{\Omega(f)-1}) := \pi\left(f^{\varepsilon}_*\left(\pi^{-1}(m_0), \cdots, \pi^{-1}(m_{\Omega(f)-1})\right)\right). \tag{3.88}$$

This defines a grammar that is compositional. □

Notice that most interesting languages fail to be monophone. Hence the notions based on string grammars are not as interesting as they appear. A more interesting notion is provided by restricting the set of grammars to weakly 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 know when they should be undefined.

Definition 3.10 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 CFGs are strongly autonomous, so the distinction between strong and superstrong turns on the possibility to have a weakly compositional or compositional CFG, respectively.

Example 3.13 (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 $A \to Bx$, $A \to \varepsilon$, or $A \to x$, A and B nonterminals and x a terminal symbol. G is **right regular** if it only has rules of the form $A \to xB$, $A \to \varepsilon$ or $A \to x$, 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: $A := \{0, \mathsf{L}\}$. For nonempty $\vec{x} \in A^*$ we put

$$\begin{aligned} n(0) &:= 0 \\ n(\mathsf{L}) &:= 1 \\ n(\vec{x}0) &:= 2n(\vec{x}) \\ n(\vec{x}\mathsf{L}) &:= 2n(\vec{x}) + 1 \end{aligned} \tag{3.89}$$

Finally,

$$L := \{\langle \vec{x}, n(\vec{x})\rangle : \vec{x} \in A^+\}. \tag{3.90}$$

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 (couched as a grammar). $F := \{f_0, f_1, f_2, f_3\}$, $\Omega(f_0) = \Omega(f_1) = 0$, $\Omega(f_2) = \Omega(f_3) = 1$.

$$\begin{aligned} \mathcal{I}(f_0)() &:= \langle 0, 0\rangle \\ \mathcal{I}(f_1)() &:= \langle \mathsf{L}, 1\rangle \\ \mathcal{I}(f_2)(\langle \vec{x}, n\rangle) &:= \langle \vec{x}^\frown 0, 2n\rangle \\ \mathcal{I}(f_3)(\langle \vec{x}, n\rangle) &:= \langle \vec{x}^\frown \mathsf{L}, 2n + 1\rangle \end{aligned} \tag{3.91}$$

There is however no independent right 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

$$f^\varepsilon(\langle \vec{x}, n\rangle) = \vec{y}^\frown \vec{x} \tag{3.92}$$

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

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

$$\mathsf{L}0, \mathsf{L}00, \mathsf{L}000, \mathsf{L}0000, \cdots \tag{3.93}$$

These strings must be obtained by adding /L/ to a string consisting in 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 $\mathcal{I}(f)$ creates two 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^μ is independent of the exponent. However, we must now have $f^\mu(0) = 8$, as well as $f^\mu(0) = 128$, a contradiction.

$$\begin{array}{lll} \mathcal{I}(f)(\langle 000, 0\rangle) & = \langle \mathsf{L}000, 8\rangle & = \langle f^\varepsilon(000), f^\mu(0)\rangle \\ \mathcal{I}(f)(\langle 0000000, 0\rangle) & = \langle \mathsf{L}0000000, 128\rangle & = \langle f^\varepsilon(0000000), f^\mu(0)\rangle \end{array} \tag{3.94}$$

⚽

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 this 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 called weakly CF if the associated string language is CF. It is called CF if there is an independent c-grammar for it whose string *and* category part taken together is CF.

Example 3.14 We continue Example 3.13. Given the same language L we show that there is no independent right regular c-language L' whose projection to $A^* \times M$ is L. This is to say, allowing *any* classification L' of string-meaning pairs into finitely many categories, there is no independent right regular c-grammar for L'. The argument is basically the same. We look at unary functions. If f is unary, it has the form

$$\mathcal{I}(\langle \vec{x}, \gamma, n\rangle) = \langle f_*^\varepsilon(\vec{x}), f_*^\kappa(\gamma), f_*^\mu(n)\rangle \tag{3.95}$$

for some f_*^ε, f_*^κ and f_*^μ. Furthermore, $f_*^\varepsilon(\vec{x}) = \vec{y}^\frown\vec{x}$. Look at the signs $\sigma_p := \langle \mathsf{L}0^p, \gamma_p, 2^p\rangle$ ($p \in \mathbb{N}$). Let t_p be an analysis term of σ_p. Either $t_p = f$ for some zeroary f, or $t_p = fs_p$ for some unary f. In the latter case, $f_*^\varepsilon(\vec{x}) = \mathsf{L}0^{k\frown}\vec{x}$ for some k that depends only on f and so s_p unfolds to $\langle 0^{p-k}, \delta_p, 0\rangle$. Now we look at f_*^μ. We have $f_*^\mu(0) = 2^p$. It follows that if $q \neq p$ then t_q does not have the form fs_q. There are however only finitely many functions. ⚽

Notice that for the argument to work we did not have to assume that there are only finitely many categories. For the argument requires only (weak!) independence of the meaning functions from the exponents and the categories.

Example 3.15 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

$$\vec{x}_0^{\frown}(\texttt{million}_{\sqcup})^{p_0}\frown\vec{x}_1^{\frown}(\texttt{million}_{\sqcup})^{p_1}\frown\cdots\frown(\vec{x}_{n-1}^{\frown}\texttt{million}_{\sqcup})^{p_{n-1}} \tag{3.96}$$

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

$$((\texttt{one}_{\sqcup} \mid \texttt{ten}_{\sqcup} \mid \texttt{one}_{\sqcup}\texttt{hundred}_{\sqcup})\texttt{thousand}_{\sqcup})? \quad (\texttt{one}_{\sqcup} \mid \texttt{ten}_{\sqcup} \mid \texttt{one}_{\sqcup}\texttt{hundred}_{\sqcup})? \tag{3.97}$$

This language is not weakly CF. It does not satisfy the Pumping Lemma (see Exercise 3.13). 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 is m-free and does not contain any occurrences of /thousand/. The grammar is given in Table 3.1. It has two modes of composition: "additive" concatenation and "multiplicative" concatenation. Since the language is unambiguous,

Table 3.1 Number names

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

we can formulate a bigrammar using string functions that are total and semantic functions that are partial. Now define

> $A(\vec{x}, \vec{y}, m, n)$ 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.

Also define

> $B(\vec{x}, \vec{y}, m, n)$ if and only if either (a) $\vec{x}$ is a block and $\vec{y} = \texttt{million}$ or (b) $\vec{x} = \texttt{one}$ and $\vec{y} \in \{\texttt{hundred}, \texttt{thousand}\}$ or (c) $\vec{x} = \texttt{one␣hundred}$ and $\vec{y} = \texttt{thousand}$. (See Fig. 3.1).

Then define the modes as follows.

$$\begin{aligned} a^{\varepsilon}(\langle \vec{x}, m\rangle, \langle \vec{y}, n\rangle) &:= \vec{x}^\frown\texttt{␣}^\frown\vec{y} \\ a^{\mu}(\langle \vec{x}, m\rangle, \langle \vec{y}, n\rangle) &:= \begin{cases} m+n & \text{if } A(\vec{x}, \vec{y}, m, n), \\ \text{undefined} & \text{else.} \end{cases} \\ m^{\varepsilon}(\langle \vec{x}, m\rangle, \langle \vec{y}, n\rangle) &:= \vec{x}^\frown\texttt{␣}^\frown\vec{y} \\ m^{\mu}(\langle \vec{x}, m\rangle, \langle \vec{y}, n\rangle) &:= \begin{cases} mn & \text{if } B(\vec{x}, \vec{y}, m, n), \\ \text{undefined} & \text{else.} \end{cases} \end{aligned} \tag{3.98}$$

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 such as /zero␣(million␣)^k/ then the semantic condition "$m > n$" in $A(\vec{x}, \vec{y}, m, n)$ must be replaced by a syntactic condition involving the number k. In this case we may however say that the semantic functions are total while the syntactic functions are restricted and so the language is not really CF. ⚽

Example 3.16 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

$$\texttt{John went to the shop not went to the shop?} \tag{3.99}$$

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

$$\vec{x}\texttt{␣}\vec{y}\texttt{␣not␣}\vec{y}\texttt{?} \tag{3.100}$$

The data for Chinese are not without problems but I shall ignore the empirical complications here and pretend that the above characterization is exact. One analysis proceeds via copying. An alternative analysis is the following. Observe that in Chinese, disjunctive statements are formed like this. To say that subject $\vec{x}$ $\vec{y}$s or $\vec{z}$s you may simply say

$$\vec{x}\texttt{␣}\vec{y}\texttt{␣}\vec{z}\texttt{.} \tag{3.101}$$

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

$$\vec{x}\sqcup\vec{y}\sqcup\mathtt{not}\sqcup\vec{z}\,. \tag{3.102}$$

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. ⚽

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 that are inherently structurally ambiguous. This means the following. Given a language L that is unambiguous, every derivation of a given exponent must yield the same meaning. Thus, as one says, all structural ambiguity is *spurious*.

Definition 3.11 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:

$$\begin{aligned} f \approx_0 g \text{ if } \Omega(f) = \Omega(g) = 0 \\ fs_0\cdots s_{n-1} \approx_0 ft_0\cdots t_{n-1} \text{ if } n > 0,\ f = g \text{ and } s_i \approx_0 t_i \text{ for all } i < n \end{aligned} \tag{3.103}$$

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 $e^\circ := \{m : \langle e, m\rangle \in L\}$.

$$L^\S := \{\langle e, e^\circ\rangle : e \in \varepsilon[L]\}. \tag{3.104}$$

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 3.17 We let $A := \{\mathtt{p}, \mathtt{0}, \mathtt{1}, \neg, \wedge, \vee\}$. $F := \{f_0, f_1, f_2, f_3, f_4, f_5\}$, $\Omega(f_0) := 0$, $\Omega(f_1) := \Omega(f_2) := \Omega(f_3) := 0$, $\Omega(f_4) := \Omega(f_5) := 2$. Meanings are sets of functions from $V := \{\mathtt{0}, \mathtt{1}\}^*$ to $\{t, f\}$. We define UBool as the language generated by the following CFG G_U. For a variable $\mathtt{p}\vec{x}$, $[\mathtt{p}\vec{x}] = \{\beta : \beta(\vec{x}) = t\}$. Given $U = [\mathtt{p}\vec{x}]$, 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.

$$\begin{aligned}
\mathcal{I}(f_0)() &:= \langle \mathtt{p}, [\varepsilon] \rangle \\
\mathcal{I}(f_1)(\langle \vec{x}, U \rangle) &:= \langle \vec{x}^\frown \mathtt{0}, [(\dagger U)^\frown \mathtt{0}] \rangle \\
\mathcal{I}(f_2)(\langle \vec{x}, U \rangle) &:= \langle \vec{x}^\frown \mathtt{1}, [(\dagger U)^\frown \mathtt{1}] \rangle \\
\mathcal{I}(f_3)(\langle \vec{x}, U \rangle) &:= \langle \neg^\frown \vec{x}, \mathrm{Val} - U \rangle \\
\mathcal{I}(f_4)(\langle \vec{x}, U \rangle, \langle \vec{y}, V \rangle) &:= \langle \vec{x}^\frown \wedge^\frown \vec{y}, V \cap U \rangle \\
\mathcal{I}(f_5)(\langle \vec{x}, U \rangle, \langle \vec{y}, V \rangle) &:= \langle \vec{x}^\frown \vee^\frown \vec{y}, V \cup U \rangle
\end{aligned} \tag{3.105}$$

Notice that this language is like natural language in being highly ambiguous: there are no brackets. Thus, the expression $/\neg \mathtt{p} \mathtt{0} \wedge \mathtt{p}/$ can be read in two ways: it has the 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. ⚽

Let us now try to see whether we can define a CFG for UBool§. We shall keep the string part of G_U from Example 3.17. Look at the strings $/\mathtt{p}\vec{x} \wedge \neg \mathtt{p}\vec{x}/$, where $\vec{x} \in \{\mathtt{0}, \mathtt{1}\}^*$. As they are uniquely readable and they have no satisfying valuation, their meaning in UBool§ is $\{\varnothing\}$. On the other hand, $/\mathtt{p}\vec{x} \wedge \neg \mathtt{p}\vec{x} \vee \mathtt{p}\vec{y}/$ has three analyses corresponding to the following bracketed strings:

$$/((\mathtt{p}\vec{x} \wedge (\neg \mathtt{p}\vec{x})) \vee \mathtt{p}\vec{y})/, /(\mathtt{p}\vec{x} \wedge (\neg (\mathtt{p}\vec{x} \vee \mathtt{p}\vec{y})))/, /(\mathtt{p}\vec{x} \wedge ((\neg \mathtt{p}\vec{x}) \vee \mathtt{p}\vec{y}))/ \tag{3.106}$$

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

$$\mathcal{J}(f_5)(\langle \mathtt{p}\vec{x} \wedge \neg \mathtt{p}\vec{x}, \{\varnothing\} \rangle, \langle \mathtt{p}\vec{y}, [\vec{y}] \rangle) = \langle \mathtt{p}\vec{x} \wedge \neg \mathtt{p}\vec{x} \vee \mathtt{p}\vec{y}, \{[\vec{y}], [\vec{x}] \cap [\vec{y}], \varnothing\} \rangle \tag{3.107}$$

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.

$$\begin{aligned}
\mathcal{J}(f_4)(\langle \mathtt{p}\vec{x}, [\vec{x}] \rangle, \langle \neg \mathtt{p}\vec{x} \vee \mathtt{p}\vec{y}, \{(\mathrm{Val} - [\vec{x}]) \cup [\vec{y}], \mathrm{Val} - ([\vec{x}] \cup [\vec{y}])\}\rangle) \\
= \langle \mathtt{p}\vec{x} \wedge \neg \mathtt{p}\vec{x} \vee \mathtt{p}\vec{y}, \{[\vec{y}], [\vec{x}][\vec{y}], \varnothing\} \rangle
\end{aligned} \tag{3.108}$$

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 CFG 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 CFGs exist. As a warm-up for the proof let us observe the following. Let φ be a formula that is composed from variables using only conjunction. Then although φ 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∧p0∧p1/, /p1∧p0∧p1∧p/, /p0∧p∧p1∧p1/. Next we consider formulae of the form $\alpha \vee \varphi$, where α is a variable and φ is of the previous form. An example is /p0∨p∧p1∧p1∧p2/. We assume that α does not occur in φ 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):

$$\begin{array}{l}(\mathtt{p0} \vee \mathtt{p} \wedge \mathtt{p1} \wedge \mathtt{p1} \wedge \mathtt{p2}) \\ (\mathtt{p0} \vee \mathtt{p}) \wedge \mathtt{p1} \wedge \mathtt{p1} \wedge \mathtt{p2} \\ (\mathtt{p0} \vee \mathtt{p} \wedge \mathtt{p1}) \wedge \mathtt{p1} \wedge \mathtt{p2} \\ (\mathtt{p0} \vee \mathtt{p} \wedge \mathtt{p1} \wedge \mathtt{p1}) \wedge \mathtt{p2}\end{array} \tag{3.109}$$

The general form is $(\alpha \vee \chi) \wedge \rho$, and its satisfying valuations make either $\alpha \wedge \rho$ or $\chi \wedge \rho$ true. α 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 χ or in ρ. How often it occurs in either of them does not matter. If v occurs twice, it may additionally occur both in χ and ρ. However, even in this case there is no difference. Assuming that v does not occur in α, χ or ρ, here are the choices if it occurs just once:

$$(\alpha \vee \chi) \wedge v \wedge \rho, (\alpha \vee \chi \wedge v) \wedge \rho \tag{3.110}$$

Here are the choices if it occurs twice:

$$(\alpha \vee \chi) \wedge v \wedge v \wedge \rho, (\alpha \vee \chi \wedge v) \wedge v \wedge \rho, (\alpha \vee \chi \wedge v \wedge v) \wedge \rho. \tag{3.111}$$

The first reading of (3.111) is the same as the first reading of (3.110), the last reading of (3.111) the same as the last reading of (3.110). The middle reading is synonymous with the first. (This argument requires χ to be nonempty.) For the purpose of the next theorem say that a bigrammar $\langle \Omega, \mathcal{I}^{\varepsilon}, \mathcal{I}^{\mu} \rangle$ is a concatenation bigrammar if $\langle \Omega, \mathcal{I}^{\varepsilon}_{*} \rangle$ 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 3.5 UBool§ *has no independent concatenation bigrammar. Hence,* UBool§ *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. We shall show that for given n there is an exponential number of formulae that 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 CFGs 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

$$\mathcal{I}(f)(\sigma_0, \cdots, \sigma_{\Omega(f)-1}) = \langle f^{\varepsilon}(e_0, \cdots, e_{\Omega(f)-1}), f^{\mu}(\sigma_0, \cdots, \sigma_{\Omega(f)-1}) \rangle. \tag{3.112}$$

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

$$V \vee (V \wedge)^+ V \vee V \tag{3.113}$$

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 is as follows, in ordinary notation:

$$\varphi = p_0 \vee p_2 \wedge p_4 (\wedge p_4) \wedge p_5 (\wedge p_5) \cdots p_{n+3} (\wedge p_{n+3}) \wedge p_3 \vee p_1 \tag{3.114}$$

Let us say that φ 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 φ; let R be a subset of I. Then by φ_R denote the formula that is like φ having a cut exactly at those i that are in R. We show first the following claim.

Claim. Let $R, S \subseteq [4, n+3] = [4, 5, \cdots, n+3]$. If $R \neq S$ then the meaning of φ_R in UBool§ is different from that of φ_S.

Let us 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 $\vee$. γ_1 may be empty; δ_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 γ, δ_1 and δ_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 δ_i and γ_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):

$$\begin{aligned} &(p_0 \vee \gamma \wedge p_i) \wedge (p_i \wedge \delta \vee p_1) \\ &= (p_0 \wedge p_i \wedge \delta) \vee (p_0 \wedge p_1) \vee (\gamma \wedge p_i \wedge p_i \wedge \delta) \vee (\gamma \wedge p_i \wedge p_1) \end{aligned} \tag{3.115}$$

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 δ_1 empty and (B) with γ_2 empty. Form (F) implies $(\neg p_0) \to \gamma$, as well as $(\neg p_1) \to \delta$. In Form (1), we therefore must have $\gamma_1 = \gamma$ and in Form (2) $\delta_2 = \delta$. Form (F) implies $\neg(p_0 \wedge p_1) \to p_i$. This is not a consequence of Forms (1) and (2), (A) or (B). Thus, Form (F) is not 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 φ_R. The immediate constituents of φ_R cannot contain two disjunction symbols. They can only contain one. In this case, however, the cuts present in φ_R are not reflected in the semantics. To conclude the argument, let us assume that the analysis term of φ_R is $f s_0 \cdots s_{\Omega(f)-1}$. We shall look at all possible analysis terms for the φ_S, $S \subseteq [4, n+3]$. We look at (3.112) 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 $p := \operatorname{card} C$ the number or 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 φ_R into maximally k^* parts of different category and different meaning. The combinations of category and meaning do not depend on R. We have

$$(2p)^{k^*}\binom{n+2}{k^*-1} < (2p(n+2))^{k^*} \tag{3.116}$$

Out of such parts we must form in total 2^n different meanings to get all the φ_S, using our modes. Assume that we have μ modes. If n is large enough, however, $\mu(2p(n+2))^{k^*} < 2^n$. □

The proof has just one gap and it consists in 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 3.14 below.)

There is an interesting further question. Consider in place of the meaning e° another one; given that meanings are propositions we can form the disjunctions of all the possible meanings.

$$\begin{aligned} e^\vee &:= \bigvee \{m : \langle e, m\rangle \in L\} \\ L^\vee &:= \{\langle e, e^\vee\rangle : e \in \varepsilon[L]\} \end{aligned} \tag{3.117}$$

This leads to the language UBool$^\vee$. It is not clear whether this language is (super)strongly CF.

Exercise 3.12 Prove Theorem 3.3. Prove that the theorem can be strengthened to languages where a string has boundedly many meanings.

Exercise 3.13 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 3.15 does not satisfy the Pumping Lemma.

Exercise 3.14 Look again at UBool. Call a **formula** a string of ε[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 3.17). 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 3.15 Use the previous exercise to show that there is no strongly independent context free grammar avoiding syncategorematic rules for UBool[§].

Exercise 3.16 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 $\mathcal{I}(f)$ have a certain property $\mathcal{P}$.

3.6 Indeterminacy in Interpreted Grammars

This section is largely based on Kracht (2008), though the proof of the central theorem has been greatly simplified. 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 3.12 Let E and M be sets of exponents and meanings, respectively. An **indeterminate interpreted grammar over** $E \times M$ is a pair $\langle \Omega, \mathcal{I} \rangle$, where Ω is a signature and for every $f \in F$, $\mathcal{I}(f) \subseteq (E \times M)^{\Omega(f)+1}$. 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 $\tau \in E \times M$:

$$\text{If for all } i < \Omega(f), \sigma_i \in S \text{ and if } \langle \sigma_0, \cdots, \sigma_{\Omega(f)-1}, \tau \rangle \in \mathcal{I}(f), \text{ then } \tau \in S. \tag{3.118}$$

This is the broadest notion, allowing to form signs from signs. Now, as before we have to replace grammars by bigrammars. The definition is completely analogous. Instead of a pair of functions f^ε and f^μ we have a pair of relations

$$\begin{aligned} f^\varepsilon &\subseteq (E \times M)^{\Omega(f)} \times E, \\ f^\mu &\subseteq (E \times M)^{\Omega(f)} \times M. \end{aligned} \tag{3.119}$$

This is called an indeterminate (interpreted) grammar. 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 $i < \Omega(f)$)

$$\text{if } \langle \vec{\sigma}, e \rangle \in f^\varepsilon \text{ then } \langle \vec{\sigma'}, e \rangle \in f^\varepsilon. \tag{3.120}$$

This can be restricted to the language generated by the grammar but we refrain from introducing too many fine distinctions. Dually, **compositionality** is defined. 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 $\langle \sigma_0, \cdots, \sigma_{\Omega(f)-1}, \langle e, m \rangle \rangle \in \mathcal{I}(f)$ and $\langle \sigma_0, \cdots, \sigma_{\Omega(f)-1}, \langle e, m' \rangle \rangle \in \mathcal{I}(f)$ 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** is defined dually.

Proposition 3.3 *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 2.22 boolean logic in Polish Notation and the unbracketed notation as given in Example 3.17. Here we shall give yet another formulation, this time with obligatory bracketing. The details are similar to those in Example 3.17. 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:

$$\begin{aligned}
\mathcal{I}(f_0)() &:= \langle \mathtt{p}, [\varepsilon] \rangle \\
\mathcal{I}(f_1)(\langle \vec{x}, U \rangle) &:= \langle \vec{x}^\frown \mathtt{0}, [\dagger(U)^\frown \mathtt{0}] \rangle \\
\mathcal{I}(f_2)(\langle \vec{x}, U \rangle) &:= \langle \vec{x}^\frown \mathtt{1}, [\dagger(U)^\frown \mathtt{1}] \rangle \\
\mathcal{I}(f_3)(\langle \vec{x}, U \rangle) &:= \langle (^\frown \neg ^\frown \vec{x}^\frown), \mathrm{Val} - U \rangle \\
\mathcal{I}(f_4)(\langle \vec{x}, U \rangle, \langle \vec{y}, V \rangle) &:= \langle (^\frown \vec{x}^\frown \wedge ^\frown \vec{y}^\frown), V \cap U \rangle \\
\mathcal{I}(f_5)(\langle \vec{x}, U \rangle, \langle \vec{y}, V \rangle) &:= \langle (^\frown \vec{x}^\frown \vee ^\frown \vec{y}^\frown), V \cup U \rangle
\end{aligned} \tag{3.121}$$

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 β and which is defined as follows.

$$\beta(\varphi) = \begin{cases} 1 & \text{if } \beta \in [\varphi], \\ 0 & \text{else.} \end{cases} \tag{3.122}$$

Example 3.18 Let B be the string language of boolean expressions. Pick a valuation β and let

$$L := \{\langle \varphi, \beta(\varphi) \rangle : \varphi \in B\}. \tag{3.123}$$

Consider an indeterminate string grammar $G = \langle F, \Omega \rangle$ for it, for example the grammar from Exercise 2.22. Put $F_2 := \{f^0, f^1 : f \in F\}$ and let $\Omega^2(f^0) := \Omega^2(f^1) := \Omega(f)$. Finally, put

$$\begin{aligned}
\mathcal{I}(f^0) := \{\langle \langle e_i, m_i \rangle : i < \Omega(f) + 1 \rangle : \langle e_i : i < \Omega(f) + 1 \rangle \in \mathcal{I}(f), \\
\beta(e_{\Omega(f)}) = 0, m_{\Omega(f)} = 0\}, \\
\mathcal{I}(f^1) := \{\langle \langle e_i, m_i \rangle : i < \Omega(f) + 1 \rangle : \langle e_i : i < \Omega(f) + 1 \rangle \in \mathcal{I}(f), \\
\beta(e_{\Omega(f)}) = 1, m_{\Omega(f)} = 1\}.
\end{aligned} \tag{3.124}$$

So the relations are split into two variants, 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 2.22, 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 /∧p∧p∧pp/. Assume that $\beta(\mathtt{p}) = 1$. Then the value of that formula is also 1. The string /∧p/ can be adjoined at several places, marked here with ∘:

$$\circ\wedge\circ\mathtt{p}\circ\wedge\circ\mathtt{p}\circ\wedge\circ\mathtt{p}\circ\mathtt{p} \tag{3.125}$$

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. ⚽

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, which 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 φ 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 3.6 *There is no independent tree adjunction bigrammar (and hence no compositional tree adjunction grammar) for* Bool *in which all meaning functions are total.*

Independence is of course essential. Since Bool is unambiguous, there can also be no compositional grammar, for autonomy can be guaranteed at no cost: the dependency of the exponents on the meanings is eliminable since we can recover the meaning from the exponent.

Before we can embark on the proof, we have to make some preparations.

Definition 3.13 Let $L \subseteq E \times M$ be an interpreted language and $D \subseteq E$. Then $L \restriction D := L \cap (D \times M)$ is the D-**fragment** of L. If $E = A^*$ and $D = B^*$ then we also write $L \restriction B$ in place of $L \restriction B^*$.

The case where we restrict to a subalphabet is the one that we shall use here. We shall study the following fragments of Bool:

$$\begin{aligned} \mathrm{Var} &:= \mathrm{Bool} \restriction \{\mathtt{p}, \mathtt{0}, \mathtt{1}\} \\ \mathrm{Bool}^{\wedge} &:= \mathrm{Bool} \restriction \{(,), \mathtt{0}, \mathtt{1}, \mathtt{p}, \wedge\} \\ \mathrm{Bool}^{\neg} &:= \mathrm{Bool} \restriction \{(,), \mathtt{0}, \mathtt{1}, \mathtt{p}, \neg\} \end{aligned} \tag{3.126}$$

Now assume G is a grammar for L. Then for every f, let

$$\begin{aligned} f^{\varepsilon} \restriction D &:= f^{\varepsilon} \restriction (D \times M) \\ f^{\mu} \restriction D &:= f^{\mu} \restriction (D \times M) \end{aligned} \tag{3.127}$$

Finally,

$$f \restriction D := (f^{\varepsilon} \restriction D) \times (f^{\mu} \restriction D). \tag{3.128}$$

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 $\sharp_a(\vec{x})$ denote the number of occurrences of a in $\vec{x}$.

For $E = A^*$, $f : E^n \to E$ is **pseudoadditive** if for every $a \in A$: either $\sharp_a(\vec{x}_i) = 0$ for all $i < n$ and then $\sharp_a(f(\vec{x}_0, \cdots, \vec{x}_{n-1})) = 0$ or

$$\sharp_a(f(\vec{x}_0, \vec{x}_1, \cdots, \vec{x}_{n-1})) \geq \sum_{i<n} \sharp_a(\vec{x}_i). \tag{3.129}$$

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 $f^{\varepsilon}(\vec{x}_0, \cdots, \vec{x}_{n-1})$. Hence we have a grammar

$$\begin{aligned} (\mathcal{I} \Rsh B)(f) &:= \mathcal{I}(f) \Rsh B \\ G \Rsh B &:= \langle \Omega, \mathcal{I} \Rsh B \rangle \end{aligned} \tag{3.130}$$

Now, $G \Rsh B$ 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 $\iota_{G \Rsh B}(t) = \iota_G(t)$. It follows that $G \Rsh B$ generates *exactly* $G \Rsh B$.

Proposition 3.4 *Suppose that G is an additive compositional bigrammar for L. Then $G \Rsh B$ is an additive compositional bigrammar for $L \Rsh B$.*

Thus if G is an adjunction grammar so is $G \Rsh B$.

Example 3.19 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

$$\mathtt{p000000\cdots0} \tag{3.131}$$

For simplicity we fix the adjunction sites to be of the form $\langle \mathtt{p}\vec{x}, \vec{y}, \varepsilon \rangle$. Depending on $\vec{x}$ we get a different variable. Thus, for any given rule only one of the adjunction sites from $\{\langle \mathtt{p0}^m, \mathtt{0}^{k-m}, \varepsilon \rangle : m \leq k\}$ 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 $\langle \mathtt{p}, \vec{y}, \varepsilon \rangle$. ⚽

Example 3.20 Another place where caution needs to be exercised when doing adjunction is the following. Let φ be a formula consisting in variables and their negations. Suppose that φ contains a variable and its negation, as in

$$\mathtt{(p01\wedge(\neg p01))} \tag{3.132}$$

Then no valuation satisfies φ. In other words, we have $\langle \varphi, \varnothing \rangle \in \text{Bool}$. 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:

$$(\mathtt{p101} \wedge (\neg \mathtt{p01})) \tag{3.133}$$

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, $\text{card}(\{f^{\mu}(\varnothing) : f \in G\}) \leq k$. 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, their adjunction is actually syntactically heavily restricted, see Kracht (2008).

Let me now prove the central theorem. Assume that we have an independent adjunction bigrammar G for $\text{Bool}^{\wedge}$. Let ρ be the number of rules of G and κ be the maximum number of symbol occurrences added by any rule. A tree is called **binary** if it only contains occurrences of /0/ and /1/. Choose a formula of the following form.

$$\varphi = (\mathtt{p}\vec{x}_0 \wedge (\mathtt{p}\vec{x}_1 \wedge (\mathtt{p}\vec{x}_2 \cdots \wedge \mathtt{p}\vec{x}_{2\rho+2}) \cdots))) \tag{3.134}$$

The length of the $\vec{x}_i$ is subject to the following restriction. (a) $|\vec{x}_i| > (2\rho + 3)\kappa$ and (b) for $i < j < 2\rho + 3$: $||\vec{x}_i| - |\vec{x}_j|| > \kappa$.

Let φ be derived by G. Then it contains at most $2\rho + 2$ occurrences of trees with symbols other than /0/ and /1/. (It is not hard to see that for every occurrence of /p/ one occurrence of /∧/, of /(/ and /)/ must be added as well and similarly for the other nonbinary symbols.) Thus, by Condition (a), each of the $\vec{x}_i$ contains occurrences added by a binary tree. Thus, in each of the variables we can somewhere adjoin a binary tree. There are $2\rho + 3$ variables. As a single adjunction can manipulate up to two variables, we have $\rho + 1$ different adjunction sites for binary trees, each manipulating a different set of variables. As we have ρ many rules, two of the adjunction sites must yield the same output semantically. (At this point totality enters; for it says that whenever adjunction is syntactically licit there is a corresponding semantic output.) Hence two of them must yield the same syntactic output. Now, adjunction at $\vec{x}_i$ can only enlarge the index by κ many symbols, which by Condition (b) does not make it the same length as any other $\vec{x}_j$, for $j \neq i$. Thus the sets of variables obtained by adjoining at different sites are different. So is their semantics. We have $\rho + 1$ sites and at most ρ different results. Contradiction.

Example 3.21 I give a letter by letter translation of Bool into English:

$$\begin{aligned} t(\mathtt{p}) &= /\mathtt{Jack\ sees\ a\ boy}/ \\ t(\mathtt{(}) &= \varepsilon \\ t(\mathtt{)}) &= \varepsilon \\ t(\mathtt{0}) &= /\mathtt{who\ sees\ a\ girl}/ \\ t(\mathtt{1}) &= /\mathtt{who\ sees\ a\ boy}/ \\ t(\mathtt{\wedge}) &= /\mathtt{who\ sees\ no\ one\ and}/ \\ t(\mathtt{\vee}) &= /\mathtt{who\ sees\ no\ one\ or}/ \\ t(\mathtt{\neg}) &= /\mathtt{it\ is\ not\ the\ case\ that}/ \end{aligned} \tag{3.135}$$

Now define the functions s as follows.

$$\begin{aligned} s(\varepsilon) &:= /\mathtt{who\ sees\ no\ one.}/ \\ s(a^\frown \vec{x}) &:= t(a)^\frown \square^\frown s(\vec{x}) \end{aligned} \tag{3.136}$$

This gives us, for example,

$$\begin{aligned} s(\mathtt{(p0\wedge(\neg p))}) = /&\mathtt{Jack\ sees\ a\ boy\ who\ sees\ a\ girl\ who\ sees} \\ &\mathtt{no\ one\ and\ it\ is\ not\ the\ case\ that} \\ &\mathtt{Jack\ sees\ a\ boy\ who\ sees\ no\ one.}/ \end{aligned} \tag{3.137}$$

Consider the set $B := \{j\} \cup \{b\vec{x} : \vec{x} \in (\mathtt{0} \mid \mathtt{1})^*\} \cup \{g\vec{x} : \vec{x} \in (\mathtt{0} \mid \mathtt{1})^*\}$. Here j is Jack, $b\vec{x}$ is the boy number $\vec{x}$ and $g\vec{x}$ the girl number $\vec{x}$. Let $U \subseteq (\mathtt{0} \mid \mathtt{1})^*$. Define $R(U)$ as follows.

$$R(U) := \begin{cases} \{\langle b\mathtt{0}\vec{x}, g\vec{x}\rangle : \vec{x} \in (\mathtt{0} \mid \mathtt{1})^*\} \\ \cup \{\langle g\mathtt{0}\vec{x}, g\vec{x}\rangle : \vec{x} \in (\mathtt{0} \mid \mathtt{1})^*\} \\ \cup \{\langle b\mathtt{1}\vec{x}, b\vec{x}\rangle : \vec{x} \in (\mathtt{0} \mid \mathtt{1})^*\} \\ \cup \{\langle g\mathtt{1}\vec{x}, b\vec{x}\rangle : \vec{x} \in (\mathtt{0} \mid \mathtt{1})^*\} \\ \cup \{\langle j, b\vec{x}\rangle : \vec{x} \in U\} \end{cases} \tag{3.138}$$

What can be shown is that the translation of $/\mathtt{p}\vec{x}/$ 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 3.6 to English. ⚽

Exercise 3.17 Recall the definition of $G_\times$ and $G^\times$ from page 65. Extend these definitions to indeterminate grammars. Construct an indeterminate grammar G for which $(G^\times)_\times \neq G$.

Exercise 3.18 Write a compositional adjunction grammar for Var.

Exercise 3.19 Let G be additive. Show that if $\iota_G(t) \in (B^* \times M)$ then $\iota_{G\restriction B}(t) = \iota_G(t)$.

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 *realizations*. As de Saussure already 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 $\langle /\texttt{mountain}/, \lambda x.\mathsf{mountain}'(x)\rangle$ is the only thing that can be said to belong to *langage* as de Saussure defined it, 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 $\lambda x.\mathsf{mountain}'(x)$. In both cases the first is the concrete entity the second the abstract one, the one that is part of language. The picture in Fig. 3.5 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 contrasting items was irrelevant.

This view was applied to both phonology and semantics. It was thought that nothing matters to linguistics beyond the contrast, or feature, itself. It would then

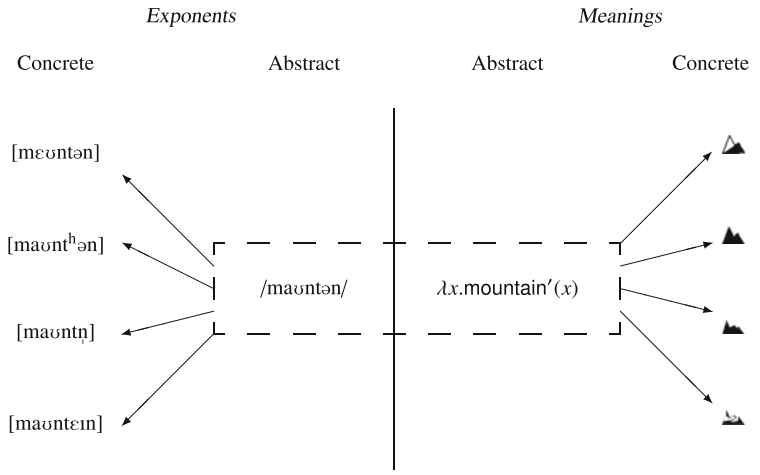

Fig. 3.5 Abstract signs

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 devoice 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 realization of abstract units. The contrasts are a matter of the underlying abstract language and they get projected to the surface via realization 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 (semantic 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; these are 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 object h, say a house, can be referred to by a particular utterance h', e. g., 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 in 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 (as are installed in GPS systems, dialog systems, trains 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 h' can be segmented into a sequence of more primitive utterances, which are instances of some sound bits corresponding to the basic sounds of English. Ideally, they correspond to the sounds [h], the diphthong [aʊ] and [s]; or maybe the diphthong is disected into [a] and [ʊ]. 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.) is 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 denote 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

$$[u]_1 := \{u' : u' \sim_L u\}, \qquad [r]_1 := \{r' : r' \sim_L r\}. \tag{3.139}$$

Finally, put

$$L_1 := \{\langle [u]_1, [r]_1\rangle : \langle u, r\rangle \in L\}. \tag{3.140}$$

Proposition 3.5 *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. □

We can formalise this as follows.

Definition 3.14 Let U and R be sets, $L \subseteq U \times R$ a language. Let $f : U \to V$ and $g : R \to S$ 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 that 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 $u \odot p \odot u'$:

$$\text{If } u \odot p \odot u', u \odot p' \odot u' \in U \text{ then } u \odot p \odot u' \sim_L u \odot p' \odot u'. \tag{3.141}$$

This can be phrased mathematically as follows: $\approx_L$ is the largest weak congruence on $\langle U, \odot\rangle$ that 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 this 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 /∅/ 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 /∅/ 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 meanings to all the four parts; furthermore, we shall assume that /good/ and /bett/ have identical meaning and with this 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 3.15 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 $\mathrm{dom}(\varphi) = \mathrm{dom}(\eta)$ and $\mathrm{dom}(\psi) = \mathrm{dom}(\theta)$,
② there is a bijection $i : L_1 \to L_2$ such that $\eta \times \theta = i \circ (\varphi \times \psi)$.

Put $U = \mathrm{dom}(\varphi)$ and $R = \mathrm{dom}(\psi)$. Then we have the following situation.

$$\begin{array}{ccc} U \times R & \xrightarrow{\varphi \times \psi} & L_1 \\ & \searrow^{\eta \times \theta} & \downarrow i \\ & & L_2 \end{array} \tag{3.142}$$

By definition there is an inverse map $j : L_2 \to L_1$. Finally, given a grammar $G = \langle \Omega, \mathcal{I}\rangle$ for $L = E \times M$ and an abstraction $\mathcal{A} = \langle \varphi, \psi, L'\rangle$ we can define the abstracted grammar $G/\mathcal{A} := \langle \Omega, \mathcal{I}^{\mathcal{A}}\rangle$ 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 σ. Then for a function symbol f define

$$\mathcal{I}^{\mathcal{A}}(f)\left(\sigma_0^{\mathcal{A}}, \cdots, \sigma_{\Omega(f)-1}^{\mathcal{A}}\right) := (\mathcal{I}(f)(\sigma_0, \cdots, \sigma_{\Omega(f)-1}))^{\mathcal{A}}. \tag{3.143}$$

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 3.22 Here is an instructive example. Suppose

$$L = \{\langle \mathsf{a}, m\rangle, \langle \mathsf{b}, m\rangle, \langle \mathsf{c}, p\rangle, \langle \mathsf{ac}, n\rangle, \langle \mathsf{bc}, n'\rangle\}. \tag{3.144}$$

The grammar consists in the following operations:

$$\begin{aligned} \mathcal{I}(f_0)() &:= \langle \mathsf{a}, m\rangle \\ \mathcal{I}(f_1)() &:= \langle \mathsf{b}, m\rangle \\ \mathcal{I}(f_2)() &:= \langle \mathsf{c}, p\rangle \\ \mathcal{I}(f_3)(\langle e, m\rangle, \langle e', m'\rangle) &:= \begin{cases} \langle \mathsf{ac}, n\rangle & \text{if } e = \mathsf{a}, e' = \mathsf{c}, \\ \langle \mathsf{bc}, n'\rangle & \text{if } e = \mathsf{b}, e' = \mathsf{c}, \\ \text{undefined} & \text{else.} \end{cases} \end{aligned} \tag{3.145}$$

/a/ and /b/ are L-equivalent. Put

$$L_1 = \{\langle \alpha, m\rangle, \langle \gamma, p\rangle, \langle \alpha\gamma, n\rangle, \langle \alpha\gamma, n'\rangle\}. \tag{3.146}$$

Let $\varphi : \mathsf{a}, \mathsf{b} \mapsto \alpha$, $\mathsf{c} \mapsto \gamma$ 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 $\mathcal{I}^{\mathcal{A}}(f_3)(\langle \alpha, m\rangle, \langle \gamma, p\rangle)$ must be both $\langle \alpha\gamma, n\rangle$ and $\langle \alpha\gamma, n'\rangle$. ⚽

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, (3.143) is a proper definition only if the congruence induced by φ and ψ is strong. Formally, the congruence induced by an abstraction is $\theta_{\mathcal{A}}$, where

$$\langle x, y\rangle \; \theta_{\mathcal{A}} \; \langle u, v\rangle \quad :\Leftrightarrow \quad \varphi(x) = \varphi(u) \text{ and } \psi(y) = \psi(v). \tag{3.147}$$

However, the condition is far too strong to be useful. A far more interesting case is when the congruence $\theta_{\mathcal{A}}$ is only weak. In this case the function is not independent of the choice of representatives; however, it is only weakly dependent. We will then say that $\mathcal{I}^{\mathcal{A}}(f)$ is simply the image of $\mathcal{I}(f)$ under φ and ψ. Then in place of (3.143) we say that $\mathcal{I}^{\mathcal{A}}(\vec{\sigma})$ is defined if there are τ_i, $i < \Omega(f)$, such that $\tau_i \; \theta_{\mathcal{A}} \; \sigma_i$ for all $i < \Omega(f)$ and $\mathcal{I}(f)(\vec{\tau})$ is defined. And in that case

$$\mathcal{I}^{\mathcal{A}}(f)\left(\sigma_0^{\mathcal{A}}, \cdots, \sigma_{\Omega(f)-1}^{\mathcal{A}}\right) := (\mathcal{I}(f)(\tau_0, \cdots, \tau_{\Omega(f)-1}))^{\mathcal{A}}. \tag{3.148}$$

Otherwise $\mathcal{I}^{\mathcal{A}}(\vec{\sigma})$ is undefined.

Example 3.23 There are two sounds in the phoneme /ɪ/, namely the voiced [ɪ] and the voiceless [ɪ̥]. They are mapped onto the same phoneme via φ. 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, $\varphi([\text{ɪ}]) = \varphi([\text{ɪ̥}])$. Thus, congruences associated with the standard phonemicization maps are generally only weak congruences. ⚽

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 one 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 phonemicization. For it also turns out that the definition requiring the substitutability of *single* occurrences is also not enough if we have weak congruences.

Example 3.24 Let $L := \{\langle \mathsf{aa}, m\rangle, \langle \mathsf{bb}, m\rangle\}$. In this situation it seems justified to postulate a single phoneme α with $\varphi(\mathsf{a}) = \varphi(\mathsf{b}) = \alpha$. The test that 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 $L' := \{\langle \mathsf{aa}, m\rangle, \langle \mathsf{bb}, n\rangle\}$. 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 {ä, e, i, ö, y} or from {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 it may be enough to exclude those phonemicizations that are improper for the other words too.)