The formalization of OT Syntax in the LFG framework

In sec. 3.3.5, the conceptual advantages of a non-derivational framework for OT syntax (and OT systems in general) were discussed. A strictly non-derivational framework for OT syntax on a formally rigid basis was first proposed by Bresnan 1996, 2001a, 2000. In (Bresnan, 2000, sec. 2), Bresnan presents a relatively close reconstruction of Grimshaw’s 1997 OT system with the formal tools of LFG, providing further arguments for the non-derivational approach.

In this chapter, I introduce an LFG-based formalization along Bresnan’s lines, discussing choices in the exact specification against the background of the general empirical and learning issues of chapter 3.

There may be various reasons for adopting the LFG formalism as the basis for an OT account of syntax, including for instance the fact that a lot of typological research has been conducted in the LFG framework. For the present purposes, the main advantage of picking LFG as the base formalism is however that its formal and computational properties have undergone thorough research and that there are highly developed systems for processing the formalism. In fact, one might say that one goal in developing the OT-LFG model is to arrive at a sufficiently restricted formalism for OT syntax in general to allow computational processing—in much the same way as the design of the LFG formalism was guided by linguistic and computational objectives: finding a framework that is expressive enough for an explanatory linguistic theory, but which at the same time guarantees that the processing tasks for grammars in that formalism are computationally tractable.
The formalization of OT Syntax in the LFG framework

It should be noted that there is a fairly rich literature on the formalization and computational properties of OT phonology: Ellison (1994), Frank and Satta (1998), Karttunen (1998), Hammond (1997), Eisner (1997), Gerdemann and van Noord (2000), Jäger (2002c,b), and others. However, the approach adopted by these researchers is based on regular languages and rational relations, which are not expressive enough for modelling the syntactic domain.  

4.1 Background on Lexical-Functional Grammar

In this section, I present a brief review of the most important formal concepts of Lexical-Functional Grammar (LFG). For more details the reader is referred to the papers in (Dalrymple et al., 1995) (for aspects of the formalism) and to the recent textbooks (Bresnan, 2001b, Dalrymple, 2001, Falk, 2001).

LFG is a non-derivational, monostratal paradigm for the representation of grammatical knowledge, first defined in Kaplan and Bresnan (1982). The key idea is to assume a number of different mathematical objects for the formal representation of different dimensions of linguistic utterances—most centrally the categorial dimension, represented through the phrase structure trees of c-structure, and the functional dimension, represented through the directed graphs of f-structure.  

As a simple example, the c-structure and f-structure for the prepositional phrase with friends is given in (64) and (65).

---

37 An interesting way of generalizing the results from formal/computational OT phonology to syntax is the generalization from regular languages to regular tree languages, as in Warten (2000) (compare also Jäger (2002c)). However, following the thread of computational-linguistic work on LFG and related formalism (as I do in this book) has the advantage that much theoretical and implementational work is already in place and can be applied with little need for adjustments.

38 (Kaplan, 1995, 11) characterizes f-structures as (hierarchical) finite functions; the graph model may however be more intuitive to readers familiar with work on unification. In the graph, identity of the values of two features (possibly under different paths) means that there is a single node, to which two feature arcs are pointing.

39 The categories are indexed in this example to suggest that we are dealing with particular instances of the category types PP, P, NP, and N.
4.1 Background on Lexical-Functional Grammar

(64) **C-structure**

```
PP

P₁ NP₃

with N₂ 'friends'
```

(65) **F-structure**

```
PRED OBJ

'with'

PRED NUM

'friend'

PLURAL
```

The arcs in the f-structure graph are labelled with features (PRED, OBJ and NUM); the atomic (i.e., leaf) f-structures are constants ('with', 'friend' and PLURAL). The common notation for f-structure graphs is as an attribute value matrix, like in (66).

(66) 

```
PRED 'with'

OBJ

PRED 'friend'

NUM PLURAL
```

The elements of the different structures stand in a correspondence relation, in the case of c- and f-structure a function \( \phi \) mapping c-structure category nodes to f-structures. This function is often called a projection. In the example, \( \phi \) maps both P₁ and PP₄ to \( f_1 \); \( f_2 \) is the \( \phi \) image for both N₂ and NP₃. Note that the function \( \phi \) need not be one-to-one, nor onto.⁴⁰ Sometimes, the function \( \phi \) is shown explicitly in the representations, using arrows:

---

⁴⁰Examples for f-structures that are not the image of any c-structure node are those representing arbitrary subjects like the subject of ‘read’ in the LFG analysis of sentences like *reading her book is fun*. In LFG, one does not assume a phonologically empty c-structure node for such a sentence.
The formalization of OT Syntax in the LFG framework

(67) PP
\[ P \rightarrow \text{with} \]
\[ NP \rightarrow \text{friends} \]

As the system is non-derivational it is important to note that one should think of all structures as coming to existence simultaneously. Projection from c-structure to f-structure should not be seen as a process; neither c-structure nor f-structure is prior to the other.

Knowledge about legal linguistic representations is formulated through propositions in a description language. For the different formal objects (trees and directed graphs), different description languages exist: the trees of c-structure are described by a context-free grammar, the f-structure graphs by formulae of a feature logic—by so-called f-descriptions. A valid f-structure is defined as the minimal model satisfying all f-descriptions (plus the additional well-formedness conditions of Completeness and Coherence, which I will discuss briefly below).

(68) Context-free grammar as a description of c-structure
\[ \text{PP} \rightarrow P \ NP \]
\[ \text{NP} \rightarrow N \]
Lexicon entries:
with P
friends N

(69) Feature logic formulae as a description of f-structure
\[ (f_1 \text{PRED}) = \text{‘with’} \]
\[ (f_1 \text{OBJ}) = f_2 \]
\[ (f_2 \text{PRED}) = \text{‘friend’} \]
\[ (f_2 \text{NUM}) = \text{PLURAL} \]

Obviously, a list of equations is interpreted conjunctively. But the feature description language contains also other Boolean connectives, so

\[ \text{To be precise, a generalization over context-free grammars is used: the right-hand side of the production rules is not defined as a string of non-terminals/terminals, but as a regular expression over non-terminals/terminals. For each language generated there exists a weakly equivalent context-free language. The advantage of allowing regular expressions in the productions is that optionality etc. can be expressed without loss of generalization, as the following rule illustrates: VP} \rightarrow V \ (NP) \ PP^* \]

60
4.1 Background on Lexical-Functional Grammar

A formula can be negated, two formulae can be connected disjunctively etc.\textsuperscript{42}

The correspondence relation between the structural dimensions is also expressed in the description language. Since the relation between c- and f-structure is a function, functional terms can be used to denote the f-structure corresponding to a given c-structure node. Thus, \( \phi(PP_4) \) refers to the f-structure projected from PP\(_4\), i.e., \( f_1 \); \( \phi(NP_3) \) refers to \( f_2 \), etc. To allow for generalized statements about the c-/f-structure correspondence we want to be able to formulate schemata for f-descriptions, relative to a particular c-structure node. Such schemata can be attached to their category nodes (this is sometimes shown by writing the descriptions above the node in the tree representation). In the schemata, a metavariable * can be used, which is instantiated to the particular c-structure node that the description is attached to.

\( \phi(*) \) denotes the f-structure projected from the node that the f-description is attached to. For expressing the relation between the f-structures projected from different nodes, we also need to be able to refer to the mother of the current node; this is done by \( \mathcal{M}(*) \). Again, \( \phi(\mathcal{M}(*)) \) denotes the f-structures projected from the mother node. So we can stipulate that the f-structures projected from the current node and its mother node are identical by stating \( \phi(*) = \phi(\mathcal{M}(*)) \) (used for both P\(_1\) and for N\(_3\)). The f-description attached to NP\(_3\) stipulates that the f-structure projected from this NP is the value of the feature OBJ in the mother node’s f-structure.

\[
(70) \quad PP_4 \quad \begin{array}{c}
\phi(*) = \phi(\mathcal{M}(*)) \\
P_1 \\
\text{with} \\
\phi(*) = \phi(\mathcal{M}(*)) \\
NP_3 \\
\text{friends} \\
N_2
\end{array}
\]

Since \( \phi(*) \) and \( \phi(\mathcal{M}(*)) \) are used extensively, there are abbreviations for these expressions, which are also called metavariables (for

\textsuperscript{42}Apart from (defining) equations, there exist other primitive types of formulae: existential constraints about feature paths, which are true when the path exists (through some other defining equation); constraining equations, which are true when a feature bears a particular value (again, defined elsewhere); and set membership constraints.\n
61
The formalization of OT Syntax in the LFG framework

f-structures): ↓ and ↑. As becomes clear when the f-descriptions are written above the nodes, the ↓ symbol is mnemonic for the (f-structure projected from the) current node—the arrow points to it. Likewise ↑ is mnemonic for the (f-structure projected from the) mother node:

![Diagram](image)

(71)

The f-descriptions originating from the lexicon (or more generally, representing morpholexical information) are typically written below the phonological/orthographic form shown as the leafs of the c-structure tree. Nevertheless, ↑ is used to denote the preterminal nodes in the tree representation (P₁ and N₂ in the example). The phonological/orthographic forms—“with”, “friends”—should not be seen as full syntactic objects, but rather as indications of the phonological properties of the syntactic words, represented as P₁ and N₂.

With these means of expression, all grammatical knowledge can be encoded in a context-free grammar with feature annotations (or short: f-annotations). The annotations are schemata for f-descriptions to be attached to the category nodes in the c-structure tree. In the rule specification, f-annotations are typically written below the category that they are attached to. (72) shows the LFG grammar and lexicon required to analyze our example PP.

(72)  

**LFG grammar: Context-free grammar with f-annotations**

\[
\begin{align*}
PP & \rightarrow P \quad NP \\
\text{↑=} & \text{↓} \\
NP & \rightarrow N \\
\text{↑=} & \text{↓}
\end{align*}
\]
4.1 Background on Lexical-Functional Grammar

**Lexicon entries:**

- with P * (↑PRED)=‘with’
- friends N * (↑PRED)=‘friend’
  (↑NUM)=PLURAL

As mentioned briefly above, there are two additional conditions on the well-formedness of f-structures: Completeness and Coherence. These ensure that the subcategorization frame introduced by verbs, prepositions and other lexical categories is actually filled in syntactically. For example John devoured fails to satisfy Completeness, while John yawned a car is ruled out by Coherence. Technically this is achieved by specifying the selected governable grammatical functions (SUBJ, OBJ etc.) in the PRED value of verbs, prepositions etc., so we would have (↑PRED)=‘with((↑OBJ))’ and (↑PRED)=‘devour((↑SUBJ) (↑OBJ))’. The composite PRED values may be seen as abbreviatory for feature structures as in (73).

(73) **Composite PRED values**

<table>
<thead>
<tr>
<th>PRED</th>
<th>OBJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>with ((↑OBJ))</td>
<td>friend</td>
</tr>
<tr>
<td>PRED 'friend'</td>
<td>NUM PLURAL</td>
</tr>
</tbody>
</table>

The expanded feature structure encodes the functor/argument structure of the predicate and forms the interface to a conceptual representation. The arc indicates that the value of the feature path PRED ARGUMENT1 and OBJ are the same f-structure object. The two values are then said to be re-entrant or structure shared (compare footnote 38).

Based on these composite PRED values, Completeness and Coherence can be formulated as follows:

---

43Under a more generalized account, only the semantic arguments are specified and the choice of particular grammatical functions is derived through mapping principles (Bresnan, 2001b, ch. 14).

44Implementations of LFG parsers like the Xerox Grammar Writer’s Workbench Kaplan and Maxwell (1996) and the Xerox Linguistic Environment (XLE; http://www.parc.xerox.com/istl/groups/alt/xle/) implement composite PRED values along these lines.
The formalization of OT Syntax in the LFG framework

(74) Completeness
All arguments specified in a predicate's subcategorization frame are also realized in this predicate's f-structure.

(75) Coherence
Only those governable grammatical functions are realized in a predicate's f-structure that are specified as arguments in the predicate's subcategorization frame.

One further peculiarity about PRED values should be noted. Their values are by definition interpreted as instantiated symbols. This means that even if the same predicate, say 'friend', is introduced twice in a sentence, the two instances will be distinct f-structure objects 'friend'₁ and 'friend'₂ and thus cannot be unified. This reflects the resource sensitivity of language and excludes that arguments are doubled, inserting them simultaneously in several c-structural position where they may occur alternatively. An example would be the following ungrammatical German sentence with the subject both in the preverbal Vorfeld position, and in the Mittelfeld:

(76) *Mein Freund ist heute mein Freund angekommen
    my friend is today my friend arrived

The language generated by an LFG grammar can be defined as the set of c-structure/f-structure pairs, such that the c-structure is generated by the context-free grammar and the f-structure is the corresponding minimal model for the f-descriptions, satisfying Completeness and Coherence. There is however one proviso in the definition (Kaplan and Bresnan, 1982, 266): Only those c-structures are considered which do not contain recursive non-branching dominance chains, as illustrated in (77).

(77) Non-branching dominance constraint/Offline Parsability
* XP
  YP
  XP

This restriction, commonly known as offline parsability, ensures that for a given string there are only finitely many c-structure analyses and thus the parsing task for LFG grammars is decidable (i.e., a procedure can be devised for the task which terminates after a finite number of steps). Quite obviously, since without this restriction a context-
4.1 Background on Lexical-Functional Grammar

free grammar may predict an infinite number of analyses for a single string, a procedure constructing f-structure models from c-structures could never stop. I will not come back to processing details until chapter 6, but it is worthwhile noting at this point that the standard LFG formalism has such a built-in restriction motivated by processing considerations.\(^{45}\)

\[ \text{(78)} \]

*Language generated by an LFG grammar*

The language \( L(G) \) generated by an LFG grammar \( G \) is the set of c-structure/f-structure pairs \( \langle T, \Phi \rangle \), such that

- \( T \) is a tree generated by the context-free grammar in \( G \)—subject to offline parsability—and
- \( \Phi \) is the minimal model satisfying all f-descriptions that arise from instantiation of the f-annotation schemata in \( G \), and satisfying the Completeness and Coherence condition.

The string language \( \mathcal{L}(G) \) generated by an LFG grammar can easily be defined as the set of terminal strings derived from the c-structures in the language \( L(G) \).\(^{46}\)

It is straightforward to extend the formal model of c- and f-structure outlined in this section to include further levels of representation, such as a semantic structure projected from f-structure. If a graph structure similar to f-structure is assumed for this semantic structure, we would have further feature-logic descriptions annotated to the rules and lexicon entries.\(^{47}\) A formally equivalent way of including such a semantic representation in the LFG structures would be to assume a special fea-

---

\(^{45}\) Strictly speaking, an LFG analysis is not uniquely specified by a pair of a c-structure tree and an f-structure; several \( \Phi \) mappings may be possible. But rather than mentioning a particular \( \Phi \) as a third component every time we refer to an LFG analysis, let us assume that the labeling function for the tree categories \( A \) of \( T \) in a pair \( \langle T, \Phi \rangle \) incorporates a reference to \( \Phi(A) \).

\(^{46}\) Throughout this book I deviate from the standard terminology of formal language theory, in which my string language is simply the language. This is to avoid clumsiness when referring to the set of analyses generated by a formal grammar—which is done much more frequently than reference to the string language. From the context, the usage should generally be obvious. Furthermore I use a typographical distinction.

\(^{47}\) The situation changes if a special resource-sensitive logic is assumed to underlie semantic construction as in the glue-language approach of Dalrymple and colleagues (see Dalrymple et al., Dalrymple et al. 1993, 1997, the contributions in Dalrymple (1999), and Dalrymple (2001) as a textbook). Then the formal treatment of semantic construction is no longer a special case of f-structure construction. In the present book I cannot go into this framework, since too little is known about generation from semantic structures—a crucial building block for an OT system.
The formalization of OT Syntax in the LFG framework

ture SEMANTICS in the f-structures.\textsuperscript{48} Given this close formal relationship between the syntactic level of f-structure and the level of semantic structure, it suffices for most formal and computational purposes relevant to the present book to just consider c-structure/f-structure pairs.

4.2 Optimality-Theoretic LFG—the overall architecture

In chapters 2 and 3, some general empirical and learning issues were discussed at a rather informal level. Defining an OT system in a formal framework such as LFG will allow us to state the issues and their consequences for the formal system in a more precise way. In particular, this will permit an investigation of computational procedures to model language processing within an OT system.

In this section, an abstract specification of the components of an OT system is given, based on the formal devices of LFG introduced in sec. 4.1. This will provide the context for a more detailed discussion of the character of the input, Gen and the violable constraints in the remainder of this chapter.

4.2.1 Abstract formal specification

Candidates The candidate analyses that OT-LFG deals with are tuples of structures as known from LFG, i.e., pairs of c-structure and f-structure (and as just mentioned possibly more levels of analysis) that are in a correspondence relation. All analyses satisfy certain basic inviolable principles, which we can assume to be encoded in an LFG grammar $G_{\text{inviol}}$, thus the set of all possible candidate analyses is defined as the structures generated by this grammar $G_{\text{inviol}}$. Sec. 4.3 contains a discussion of what principles are encoded in this grammar $G_{\text{inviol}}$.

(79) Definition of possible candidates

The set of possible candidates is defined as the language $L(G_{\text{inviol}})$ generated by a formal LFG-style grammar $G_{\text{inviol}}$.

There are some issues as to how closely the LFG-style grammars used in OT-LFG systems resemble standard formal LFG grammars (see the definition of the language generated by an LFG grammar (78)). In

\textsuperscript{48}There are some technical issues I cannot go into here: in order to be able to express the sharing of information correctly and in a general way, either the overall feature geometry has to be changed, or a special restriction operator has to be assumed Wedekind and Kaplan (1993).
4.2 Optimality-Theoretic LFG—the overall architecture

sec. 4.3, I discuss whether the Completeness and Coherence conditions apply. In sec. 4.5 and chapter 6 the issue is addressed whether the offline parsability condition should apply for the LFG-style grammars used in OT-LFG systems.

Index/Input With the candidate analyses being fully specified LFG analyses, an appropriate representation for the input in the OT sense is a partially specified representation of LFG analyses. This gives us the strictly non-derivational system I argued for in sec. 3.3.5.

For recasting Grimshaw’s 1997 analysis within LFG (Bresnan, 2000, sec. 1.1) assumes as the input “a (possibly underspecified) feature structure representing some given morphosyntactic content independent of its form of expression”. An example (that in English would have I saw her as its optimal realization) is given in (80).

\[
\begin{align*}
&PRED 'see' \\
&GF_1 \\
&PRED 'PRO' \\
&PERS 1 \\
&NUM SG \\
&TNS PAST \\
&GF_2 \\
&PRED 'PRO' \\
&PERS 3 \\
&NUM SG \\
&GEND FEM \\
\end{align*}
\]

More generally, we may want to assume an input comprising a feature representation of the semantic content of an utterance (and potentially some further “pragmatic” clues, such as information structural status etc., cf. sec. 3.3.2). So, the input is defined as follows:

\[
\text{(81) \, \textbf{Definition of the input}} \\
\text{The input is a member of the set of well-formed (partial) f-structures \( \mathcal{F} \).}
\]

Note that contrary to the situation with derivational candidates assumed in some OT syntactic work (cf. the discussion in sec. 3.3.5), there is no issue at what point of a candidate derivation the input information is available. Both the candidates and the input are formal objects that we should think of as static (with information about all arguments).

\[49\text{The status of the arguments } x \text{ and } y \text{ in the semantic form } 'see(x, y)' \text{ will be discussed in sec. 4.3.2.}\]
levels being available simultaneously). The relations between them are just mathematic relations between formal objects. Two kinds of input-candidate relations are relevant for a formal OT system: (i) \textit{Gen} -- involving a relation between the input and a set of candidates --, and (ii) the faithfulness constraints—involving a relation between an individual candidate and the input.

**Candidate generation** We can now give the following general definition of the function \textit{Gen}, depending on the grammar \( G_{\text{invio1}} \):

(82) **Definition of \textit{Gen}**

\[ \text{Gen}_{G_{\text{invio1}}} \] is a function from the set of f-structures to the power set of the analyses (c-structure/f-structure pairs \( \langle T, \Phi \rangle \)) in \( L(G_{\text{invio1}}) \):

\[ \text{Gen}_{G_{\text{invio1}}}: \mathcal{F} \rightarrow \wp(L(G_{\text{invio1}})) \]

In other words, \( \text{Gen}_{G_{\text{invio1}}} \) takes each input f-structure \( \Phi_{\text{in}} \) to a set of candidate analyses, which are contained in \( G_{\text{invio1}} \):

\[ \text{Gen}_{G_{\text{invio1}}} (\Phi_{\text{in}}) \subseteq \{ \langle T, \Phi \rangle | (T, \Phi) \in L(G_{\text{invio1}}) \} \]. Further restrictions will be discussed below.

**Constraint marking** The OT constraints come into play when the alternative candidate analyses in the \( \text{Gen}_{G_{\text{invio1}}} \) image of a given input f-structure are evaluated. The function \textit{marks} assigns counts of constraint violations to each member of the candidate set. There are different ways in which the counts of constraint violations can be captured formally: in sec. 2.1, a multiset of constraint violation marks was assumed—for instance \{*C^2, *C^2, *C^4\} for a candidate violating constraint \( C^2 \) twice and constraint \( C^4 \) once (compare the example (5) in sec. 2.1). An alternative, but equivalent way is to represent the number of violations that each constraint incurs as a natural number. I will adopt this representation in this definition.

The constraints \( \mathcal{C} \) used in an OT system are given as a sequence \( \langle C^1, C^2, \ldots, C^k \rangle \). So we can represent the violation counts for a particular candidate as a sequence of natural numbers: \( \langle n^1, n^2, n^3 \ldots n^k \rangle : n^i \in \mathbb{N}_0 \). So, assuming a constraint sequence \( \langle C^1, C^2, C^3, C^4, C^5 \rangle \) for the above example, the violation counts would be \( \langle 0, 2, 0, 1, 0 \rangle \).

An individual constraint \( C^4 \in \mathcal{C} \) is then defined as a function taking a pair of an input f-structure and an LFG analysis to a natural number. The function \textit{marks} depends not only on the candidates, but on
4.2 Optimality-Theoretic LFG—the overall architecture

the input structure too, for the following reason: we not only have markedness constraints (which are defined on the candidate structures alone) but also faithfulness constraints (which are defined relative to the input).

(83)  Definition of OT constraints

\[ C : \text{a sequence of constraints } \langle C_1, C_2, \ldots, C_k \rangle; \]
for each constraint \( C_i, i = 1..k: \)
\[ C_i(\Phi_{in}, \langle T, \Phi' \rangle) \in \mathbb{N}_0 \]

The specification of markedness constraints based on LFG analyses is discussed in sec. 4.4. Faithfulness constraints are discussed in sec. 4.5.

For the function \( \text{marks}, \) which takes into account all constraints, we get:

(84)  Definition of \( \text{marks} \)

\( \text{marks}_C \) is a function from (input) f-structures and LFG analyses to a sequence of \( k \) natural numbers
(where \( k = |C|, \) the size of the constraint set)
\[ \text{marks}_C : \mathcal{F} \times L(G_{\text{struc}}) \mapsto \mathbb{N}_0^k, \text{ such that} \]
\[ \text{marks}_C(\Phi_{in}, \langle T, \Phi' \rangle) = \langle C_1(\Phi_{in}, \langle T, \Phi' \rangle), C_2(\Phi_{in}, \langle T, \Phi' \rangle), \ldots, C_k(\Phi_{in}, \langle T, \Phi' \rangle) \rangle \]

Harmony evaluation The key concept of optimization—harmony evaluation in the narrow sense—depends on the constraint violation counts for each input/candidate pair from a given candidate set \( \text{Gen}_{G_{struc}}(\Phi_{in}) \), and on the language specific constraint ranking over the constraint set \( \succsim_{\mathcal{C}} \). The function \( \text{Eval} \) formalizing this concept determines the most harmonic candidate according to definition (3), repeated here.

(3)  Candidate \( A_i \) is more harmonic than \( A_j \) \( (A_i \succ A_j) \) if it contains fewer violations for the highest-ranked constraint in which the marking of \( A^i \) and \( A^j \) differs.

---

50 In sec. 4.5 the input-dependence of \( \text{marks} \) will be eliminated, since under the definition of \( \text{Gen}_{G_{struc}} \) adopted in sec. 4.3.1 the relevant faithfulness constraints can be defined on the candidate structures alone.

69
The formalization of OT Syntax in the LFG framework

(85) **Definition of Eval**
Given a set of LFG analyses \( \Gamma \),
\[
\text{Eval}_{\langle C, \gg \ell \rangle} (\Gamma) = \{ \langle T_j, \Phi_j \rangle \in \Gamma \mid \langle T_j, \Phi_j \rangle \text{ is maximally harmonic for all analyses in } \Gamma, \text{ under ranking } \gg \ell \} 
\]

Note that this definition is compatible both with a classical, strict constraint ranking and with a stochastic constraint ranking as assumed by Boersma (1998). As discussed briefly in sec. 3.3.1, the latter model is superior from the perspective of learnability, and it also constitutes an interesting mechanism for deriving optionality and frequency effects.

**Language generated by an OT-LFG system**

An OT-LFG system is specified by three components.

(86) **Definition of an OT-LFG system**
An OT-LFG system \( \mathcal{O} \) is defined as \( \langle G_{\text{inviol}}, \langle C, \gg \ell \rangle \rangle \), where
- \( G_{\text{inviol}} \) is a formal LFG-style grammar defining the possible candidate analyses,
- \( C \) is a sequence of OT constraints, and
- \( \gg \ell \) is a ranking over \( C \).

Finally, the language generated by an OT-LFG system can be defined as the set of analyses \( \langle T_j, \Phi_j \rangle \) for which there exists an input f-structure \( \Phi_{in} \) such that \( \langle T_j, \Phi_j \rangle \) is among the most harmonic candidates for that input (i.e., \( \text{Eval}_{\langle C, \gg \ell \rangle} (\text{Gen}_{G_{\text{inviol}}}(\Phi_{in})) \)).

(87) **Definition of the language generated by an OT-LFG system**
\[
\mathcal{L}(\mathcal{O}) = \{ \langle T_j, \Phi_j \rangle \in \mathcal{L}(G_{\text{inviol}}) \mid \exists \Phi_{in} : \langle T_j, \Phi_j \rangle \in \text{Eval}_{\langle C, \gg \ell \rangle} (\text{Gen}_{G_{\text{inviol}}}(\Phi_{in})) \}
\]

The string language \( \mathcal{L}(\mathcal{O}) \) generated by an OT-LFG system can be defined accordingly as the set of terminal strings for such analyses \( \langle T_j, \Phi_j \rangle \).

An example of a linguistic OT-LFG system satisfying this definition will be developed in the subsequent sections (specifically in sec. 4.3 and 4.4) which will focus on the concrete specification of the components.

4.2.2 Degrees of freedom in this OT-LFG architecture

Based on the definitions just presented, there are three essential components specifying a particular OT-LFG system: the “base grammar”
4.2 Optimality-Theoretic LFG—the overall architecture

$G_{\text{invioL}}$, the OT constraints $\mathcal{C}$, and the constraint ranking $\succsim_{\mathcal{C}}$. The formalization leaves these components open to be filled out by empirical and conceptual linguistic work. Recall that by assumption of the linguistic OT approach, the former two are universal, and only the latter is language-specific. So, it is a goal for linguistic research to determine the specification of the former two ($G_{\text{invioL}}$ and $\mathcal{C}$) in such a way that the language-specific ranking $\succsim_{\mathcal{C}}$ is learnable based on language data.

Note however that in addition there are some further degrees of freedom in this OT-LFG architecture, on a more technical level:

(88) Degrees of freedom in the definitions (79)–(87)

a. The definition of possible candidates is based on the language generated by a “formal LFG-style grammar $G_{\text{invioL}}$”; this leaves open whether $G_{\text{invioL}}$ is interpreted as an LFG grammar in the strict sense or whether the Completeness and Coherence condition is modified and/or the offline parsability condition is loosened up.

b. The exact specification of the function $\text{Gen}_{G_{\text{invioL}}}$ is left open.

c. It has not been fixed how the constraints are formally specified, given the input and a candidate structure.

For clarification of the last two points note that the definitions (82) and (83) given above specify only what kind of function $\text{Gen}_{G_{\text{invioL}}}$ is (mapping an f-structure to a set of candidates) and what kind of function a constraint is (mapping a candidate analysis to a natural number). This leaves completely open how the functions are specified. For example, the function $\text{Gen}_{G_{\text{invioL}}}$ may be a constant function that assigns the same candidate set to all possible input f-structures; or it may assign different candidate sets to different f-structures, based on some structural relationship. Likewise for the constraints, there is a wide spectrum of possibilities: the numbers they assign to candidate structures may be based on the counting of simple structural patterns, or they may involve complicated conditions (for instance, it is conceivable to formulate a constraint as a context-free grammar, assigning 0 when the candidate c-structure is included in the language and 1 otherwise).

All three aspects listed in (88) have to be pinned down to complete the formal specification of an OT-LFG system. The choices do not seem to have as clear an empirical impact as variations of the main components of the OT-LFG system have. But (i) they do have significant impact on processing complexity of the formal system (as I show in
sec. 4.2.3), and (ii) the basic assumptions of the OT approach discussed in chapter 2 and the empirical and conceptual observations of chapter 3 may be reflected more or less adequately in the formalization, depending on these choices.

As an example for point (ii) note that a system relying on LF-unfaithful candidates as discussed in sec. 3.3.3 is compatible with the definitions. There need not be any structural connection between the input/Index that $Gen_{G_{inv}}$ takes as its arguments and the set of candidates it assigns to this input. In sec. 3.3.4, I argued that this circumstance is undesirable from the point of view of learnability.

4.2.3 Undecidability arguments for unrestricted OT systems

To see the impact of the choices in (88) on the processing complexity, let us look at two non-linguistic examples of OT systems both of which allow the construction of an undecidability argument (they can be seen as a variant of the sketch of an undecidability argument for unrestricted OT systems that Johnson (1998) presents).

The first construction creates a scheme of OT systems $O_1$ for which an effective optimization procedure could only be devised in case another problem—the emptiness of the intersection of two context-free languages (89)—could be solved. However, since problem (89) is known to be undecidable, the optimization problem for $O_1$ must be undecidable too.

(89) The emptiness problem of the intersection of two context-free languages
Given two arbitrary context-free grammars $G_1$ and $G_2$, is $L(G_1) \cap L(G_2) = \emptyset$?

Undecidability due to powerful constraints

(90) A constructed OT-LFG system (schema) $O_1$
Let $G_1$ and $G_2$ be context-free grammars.
$G_1$ has the starting symbol $S_1$.
Specify the c-structure part of $G_{inv}$ by adding the following productions to $G_1$:
$S \rightarrow S_1$
$S \rightarrow yes$,
where $S$ is a new symbol, used as the start symbol of $G_{inv}$ and yes is a new terminal symbol. (The f-annotations in $G_{inv}$ are irrelevant. We may assume arbitrary f-structures.)
4.2 Optimality-Theoretic LFG—the overall architecture

Define \( Gen_{\text{OT-LFG}} \) as follows:
\[ Gen_{\text{OT-LFG}}(\Phi) = L(G_{\text{simvol}}), \]
for all \( \Phi \)

Assume two constraints:
\[ C^1(\Phi, (T, \Phi')) = \begin{cases} 0 & \text{if the terminal string of } T \text{ is in } L(G_2) \cup \{ \text{yes} \} \\ 1 & \text{otherwise} \end{cases} \]
\[ C^2(\Phi, (T, \Phi')) = \begin{cases} 1 & \text{if the terminal string of } T \text{ is yes} \\ 0 & \text{otherwise} \end{cases} \]

Assume the ranking \( C^1 \gg_L C^2 \).

Note that \( Gen_{\text{OT-LFG}} \) is a constant function, assigning the full set of possible candidates to any input; \( C^1 \) is a constraint based on the context-free language \( G_2 \).

The system works fine if we assume simple grammars for \( G_1 \) and \( G_2 \). For example, we may assume \( L(G_1) = \{a, aa\} \) and \( L(G_2) = \{b, c\} \). The possible candidate strings generated by \( G_{\text{simvol}} \) are then \( \{a, aa, \text{yes}\} \).

Now, let us check whether the string \text{yes} is in the string language generated by the OT-LFG system. According to the definition there has to be some input such that \text{yes} is the terminal string of the optimal analysis from the candidate set assigned to that input by \( Gen_{\text{OT-LFG}} \). Since all candidate sets are the same (and the input plays no role) this is easy to check, we need only look at one tableau:

\[
\begin{array}{|c|c|c|}
\hline
\text{Input: (arbitrary)} & C^1 & C^2 \\
\hline
\text{a. } a & \ast ! & \ast ! \\
\text{b. } aa & \ast ! & \ast ! \\
\text{c. } \ast ! \text{ yes} & \ast ! & \ast ! \\
\hline
\end{array}
\]

So, \text{yes} is indeed in the string language. In fact, it is the only string in that language. If we change \( G_2 \) from the previous example to make \( L(G_2) = \{a, b, c\} \), we get the following tableau:

\[
\begin{array}{|c|c|c|}
\hline
\text{Input: (arbitrary)} & C^1 & C^2 \\
\hline
\text{a. } \ast ! \text{ a} & \ast ! & \ast ! \\
\text{b. } aa & \ast ! & \ast ! \\
\text{c. } \ast ! \text{ yes} & \ast ! & \ast ! \\
\hline
\end{array}
\]

The string \text{yes} is no longer in the language generated (instead, all strings that are in the intersection of the two languages are, since they
The formalization of OT Syntax in the LFG framework

all have the same constraint profile). The only way yes can ever win is that all other candidates fail to satisfy \( C^1 \). So what the OT-LFG system schema \( O_1 \) in (90) effectively does is check whether the intersection of two context-free languages is empty. This problem is known to be undecidable in the general case; so the recognition problem for unrestricted OT-LFG systems (checking whether a certain string is in the string language) is also undecidable in the general case.

Since the formal system used for this argument contained an unnecessarily powerful constraint, it is worthwhile to investigate more restricted definitions.

Undecidability due to powerful candidate generation

This second type of undecidability argument was suggested to me by Jürgen Wedekind (p.c.). It exploits the fact that the emptiness problem for an LFG language is known to be undecidable\(^{51}\) and uses a powerful \( G_{\text{inviol}} \) in candidate generation. This system can be constructed as follows:

(93) Another constructed OT-LFG system (schema) \( O_2 \)

Assume an LFG grammar \( G_1 \) with start symbol \( S_1 \), for which the problem \( \mathcal{L}(G_1) = \emptyset \) is undecidable.

Construct from \( G_1 \) the grammar \( G_{\text{inviol}} \) with the new start symbol \( S \), a new nonterminal symbol \( Y \) and the new terminal symbol \( \text{yes} \). The following productions are added to the productions of \( G_1 \):

\[
S \rightarrow Y
\]

\[
(\uparrow \text{PRED})='\text{yes}'
\]

\[
S \rightarrow S_1
\]

\[
(\uparrow \text{CHECK})=+
\]

\[
(\uparrow \text{PRED})='\text{yes}'
\]

\[
Y \rightarrow \text{yes}
\]

Assume a single constraint:

\[
C^1(\Phi, (T, \Phi')) =
\begin{cases} 
0 & \text{if } \Phi' \text{ has the feature } [\text{CHECK}+] \\
1 & \text{otherwise}
\end{cases}
\]

The candidate \( \text{yes} \) violates the constraint \( C^1 \) (since its f-structure does not contain the feature \( \text{CHECK}+ \)). Analyses making use of the rule \( S \rightarrow S_1 \) satisfy \( C^1 \). But we can nevertheless get \( \text{yes} \) as the optimal

\(^{51}\)According to the introduction to part V in (Dalrymple et al., 1995, 333), this was first shown in 1983 by Kelly Roach.
4.2 Optimality-Theoretic LFG—the overall architecture

candidate for \( \Phi_{in} = [\text{PRED} \ 'yes'] \): there may be no other candidates in
the candidate set. This is the case iff no candidates can be derived with
the \( S_1 \) symbol, i.e., when \( L(G_1) = \emptyset \). Since by assumption this problem
is undecidable it is also undecidable whether \( yes \in L(O_2) \).

What is unintuitive about this OT system is that there is no struc-
tural relationship between the f-structure constructed in the original
LFG grammar \( G_1 \) (i.e., \( \phi(S_1) \)) and the f-structure projected from the
new start symbol, i.e., \( \phi(S) \). \( S \) is always mapped to the f-structure
\[ \begin{array}{c}
\text{PRED} \ 'yes' \\
\text{CHECK} + 
\end{array} \]
—no matter what \( \phi(S_1) \) is like. But in a linguistic grammar, this ig-
nored part of the structure reflects the candidate’s interpretation. The
intuition that all candidates should have the same underlying interpre-
tation, modelled by the semantic part of f-structure, is not observed.

The two undecidability arguments motivate that both the express-
ive power of individual constraints and the expressive power of candidate
generation have to be suitably restricted.\(^{52}\)

4.2.4 Fixing the choices in the definitions

An objective of this book is to arrive at a formalization of OT syntax that
is sufficiently restricted to permit realistic processing, while at the same
time meeting the assumptions and intuitions underlying the theoretical
OT approach. Further details of processing complexity will not be dis-
cussed until chapter 6; the strategy I adopt is to first fix the degrees of
freedom mentioned in (88) in a way compatible with the empirical and
conceptual criteria noted in the previous chapters and show afterwards
that the resulting system is also adequate for processing.

Point (88a)—addressing the freedom of candidate generation based
on \( G_{\text{inviol}} \)—is relevant for the treatment of unfaithful candidates. I will
come back to it briefly in sec. 4.3 and sec. 4.5 and particularly in chap-
ter 6. For the moment, the decision is not of crucial relevance.

The points (88b) and (88c)—the exact specification of \( Gen_{G_{\text{viol}}} \) and
the individual constraints—appear to be antagonistic. If \( Gen_{G_{\text{viol}}} \) goes

\(^{52}\)I will discuss a third undecidability argument, related to the second one presented
here, in sec. 6.3.1: ensuring that the f-structure is considered for all candidates does not
suffice to guarantee decidability of the recognition problem for OT-LFG systems (with-
out offline parsability). F-structure must furthermore be related to the surface string, or
possibly a context representation.
The formalization of OT Syntax in the LFG framework

a long way in restricting the possible candidate structures to the observable language data, then the constraints need not be highly expressive, since they only have to control comparatively small choices. However, this goes against the methodological principle of the OT approach (25), discussed in sec. 3.2.1 and repeated here for convenience.

(25) Methodological principle of OT
Try to explain as much as possible as an effect of constraint interaction.

Prima facie this principle would suggest a maximally general and unrestricted Gen\textsubscript{Gen\textsubscript{Gen}}. But as we have seen in sec. 4.2.2 (and sec. 3.3.2), when applied in syntax, such a concept of Gen\textsubscript{Gen\textsubscript{Gen}} leads to problems both with learnability considerations and with decidability.

For the standard conception of OT phonology the learnability and decidability problem with a weak Gen\textsubscript{Gen\textsubscript{Gen}} do not pose themselves. Learnability is not negatively affected if all candidate sets are identical and just the input differs: the candidate analyses do not contain a “deep”, interpreted level of representation that may differ from the input (like LF or the f-structure in syntax); rather the interface to the rest of the linguistic and cognitive system is concentrated in the input. So inferences from the utterance context permit direct conclusions about the input.\textsuperscript{53}

The decidability problem does not arise either—if rational relations are assumed for modelling Gen\textsubscript{Gen\textsubscript{Gen}} and regular expressions for the constraints, as is assumed in computational OT phonology. Following insights of Frank and Satta (1998), the entire OT system can be implemented as a single finite-state transducer implementing Gen\textsubscript{Gen\textsubscript{Gen}} and the constraints composed by “lenient” composition (in the terminology of Karttunen (1998)).\textsuperscript{54}

\textsuperscript{53}As pointed out in footnote 29 on page 45, such inferences from utterance context may indeed be required for phonological OT systems, since surface forms can be ambiguous (cf. Hale and Reiss (1998) for the ambiguity problem posed by a simple strong bidirectional OT system not making use of information from the utterance context in the determination of the underlying form). But if the inferences work properly, they suffice to determine the underlying input. This is not guaranteed for unrestricted OT syntax systems, in which LF and the input may differ. (Note that contextual inferences can provide only information about the candidate’s actual LF, and thus not necessarily about the input which may not be faithfully rendered at LF).

\textsuperscript{54}However for multiple violations of a single constraint a fixed upper bound has to be assumed.
4.2 Optimality-Theoretic LFG—the overall architecture

However for OT syntax we need candidate structures which contain their own level of interpretation (related to the input in some way or another, let us assume here it is identical to the input). The set of meaning representations, e.g., predicate argument structures, which underlie syntax cannot be described with regular expressions, we need at least a context-free grammar. Of course, well-formedness of the input need not be checked by the OT system itself, but it has to be able to maintain the structure (for the fully faithful candidates). As a consequence, the overall OT system will be capable of generating a context-free language when fed with appropriate predicate-argument structures: if we rank faithfulness to the input structure highest, the strings generated will be based on a phrase structure isomorphic to the underlying meaning structure. In this sense the candidate generation component Gen for syntax needs to be stronger than in phonology (in terms of automata theory, a string transducer is not sufficient, but we need a tree transducer; cf. e.g., Gécseg and Steinby (1997), and Wartena (2000) for a discussion of OT syntax). The individual constraints need not be very powerful since they can make reference to the “structural skeleton” provided by the input and maintained by Gen.

Within these limits, principle (25) tells us to choose the weakest possible Gen, so the main explanatory burden is on constraint interaction. (A criterion for deciding what should definitely follow as an effect of constraint interaction is crosslinguistic comparison, as was done for the syntactic MAX-IO and DEP-IO violations in sec. 3.2.3.)

The setup I propose uses (formally) very simple individual constraints (discussed in sec. 4.4) and a conception of Gen that disallows divergences at the level of the interpreted part of the candidate structures, i.e., the part of f-structure that corresponds to the input (sec. 4.3). (Translated to the Chomskyan terminology, this covers both the predicate-argument structure at d-structure and the interpretation-relevant aspects of LF.)

Although this may sound like a massive restriction, this Gen conception is compatible with a view that maintains maximal generality at the surface level of syntax, as will be discussed in sec. 4.5.\(^{55}\)

\(^{55}\)So the stronger \(Gen_{\text{not-id}}\) restrictions in a comparison with the OT phonology model may be viewed as no more than a compensation for the different type of representations used in syntax. A totally different formalization of OT syntax might attempt to use only the surface form as candidates and use more expressive constraints, circumventing the decidability problem addressed in sec. 4.2.2 in a different way.

77
The formalization of OT Syntax in the LFG framework

4.3 Candidate generation and the inviolable principles

4.3.1 The restricted definition of $Gen_{G_{inviol}}$

Based on the considerations of sec. 4.2.4, the candidate generation function $Gen_{G_{inviol}}$ is defined as follows:

(94) Restricted definition of $Gen$

$$Gen_{G_{inviol}}(\Phi_{in}) = \{ (T, \Phi') \in L(G_{inviol}) \mid \Phi_{in} \sqsubseteq \Phi', \text{ where} \Phi' \text{ contains no more semantic information than } \Phi_{in} \}$$

So only those of the possible candidate analyses in $L(G_{inviol})$ are picked whose f-structure is subsumed by the input f-structure, and which do not add any semantically relevant information. There is some leeway for the exact definition of semantic information. But note that in every concrete formulation of $G_{inviol}$, this can be specified by declaring particular features or feature combinations as contributors to semantic information.\(^{56}\)

This formalization meets the intuition of Bresnan’s 2000 original account of candidate generation in OT-LFG: at an abstract level, $Gen_{G_{inviol}}$ is indeed generation from the input with an LFG grammar. The candidate analyses can be viewed as being generated from the input structure by monotonically adding (non-semantic) information. Conveniently, the task of computing $Gen_{G_{inviol}}(\Phi_{in})$ is then exactly the classical task of generation from an underspecified f-structure, given an LFG grammar ($G_{inviol}$). Processing is discussed further in chapter 6.

4.3.2 Completeness and Coherence in OT syntax

There is an alternative way of stating the restricting condition in candidate generation, drawing a parallel to the standard LFG concepts of Completeness (74) and Coherence (75).\(^{57}\) What we want to implement is that candidates may contain neither more nor less semantic information than specified in the input. We could assume a special place in the

---

\(^{56}\) The qualification “where $\Phi'$ contains no more semantic information than $\Phi_{in}$” could be expressed more formally as follows:

$$\Phi' \mid_{F_{sem}} \sqsubseteq \Phi_{in}$$

where $\Phi' \mid_{F_{sem}}$ is defined as the largest f-structure $\Phi' \sqsubseteq \Psi$, such that all features contained in its path are members of $F_{sem}$ (the set of semantic features).

\(^{57}\) Compare also Wedekind’s 1995 terminology in his discussion of generation.
4.3 Candidate generation and the inviolable principles

f-structure for encoding the input information, e.g., within the composite PRED values (compare (73)). The input (80) would then look roughly as follows (note that all input information has to be coded into the structures under PRED—here I use ad hoc feature names like REF-NUM for the semantic referent’s number specification etc.):

(95) Potential way of re-coding the OT input

```
<table>
<thead>
<tr>
<th>PRED</th>
<th>ARGUMENT1</th>
<th>ARGUMENT2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FUNC TOR</td>
<td>FUNC TOR</td>
</tr>
<tr>
<td></td>
<td>see</td>
<td>PRO</td>
</tr>
<tr>
<td>TNS</td>
<td>PAST</td>
<td>PRO</td>
</tr>
<tr>
<td></td>
<td>REF-PERS</td>
<td>REF-PERS</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>REF-NUM</td>
<td>REF-NUM</td>
</tr>
<tr>
<td></td>
<td>SG</td>
<td>SG</td>
</tr>
<tr>
<td></td>
<td>REF-GEND</td>
<td>FEM</td>
</tr>
</tbody>
</table>
```

(If the example contained an adjunct, this would have been included under PRED too.) Now we can express restricted candidate generation through Completeness and Coherence conditions:

(96) OT-Completeness
All information specified as a predicate’s input information (= under PRED) is also realized in this predicate’s f-structure.

(97) OT-Coherence
Only the semantic information that is specified as a predicate’s input information (= under PRED) is also realized in this predicate’s f-structure.

This would permit us to move the subsumption restriction on Gen\textsubscript{G\textsubscript{invol}} into the formal specification of the underlying LFG-style grammar, i.e., into the definition of possible candidate structures in general (79). Candidates with an over- or under-informative f-structure would be incoherent or incomplete and thus no longer exist in G\textsubscript{invol}, so they would not have to be excluded in the definition of Gen\textsubscript{G\textsubscript{invol}}.\footnote{We would get (i) Gen\textsubscript{G\textsubscript{invol}}(\Phi\textsubscript{in}) = \{ (T, \Phi') \in L(G\textsubscript{invol}) \mid \Phi\textsubscript{in} is the input information of (T, \Phi') \}}

However, I will keep to definition (94), assuming it to be more perspicuous. We may think of the OT-Completeness and OT-Coherence conditions to be (redundantly) at work within G\textsubscript{invol} nevertheless.
It is interesting to note a change in the source of the reference information for checking Completeness/Coherence. In classical LFG, this information is contributed by the lexical entries of the predicates (i.e., the verbs, prepositions etc.). The PRED value with the subcategorization frame is conveniently written in one line, using angle brackets as in \((\text{PRED})='\text{devour}'(\text{SUBJ} \text{ OBJ})'\) (compare (73)). In OT-LFG, the relevant reference information is contributed by the input. This means that if we nevertheless assume subcategorization frames in the lexicon entries then they are pretty much redundant. Only those matching the input will be usable (since they have to be unified with the input). The only interesting case arises when we look at pathological inputs like a strictly transitive verb such as devour used with just a single argument. Here, a lexicon lacking \((\text{PRED})='\text{devour}'(\text{SUBJ})'\) would lead to an empty candidate set, predicting that the input cannot be expressed in any language. It is questionable however whether such an approach would be in the spirit of OT. For instance, OT phonology will predict a surface realization for any underlying nonsense string of phonemes. The learner will just never come across such a string and thus it will not be entered into the inventory of underlying forms. Of course, the situation is slightly different in syntax, since we are not dealing with finite lists of underlying forms. However, we have to assume that the inputs are processed further in the conceptual cognitive system, and this further processing detects and excludes nonsense inputs. The syntactic system itself will be more perspicuous and general if it does not attempt to anticipate such conceptual restrictions.

In other words, it seems most adequate for the OT syntax setup to assume lexicon entries without a specification of the subcategorization frame, compatible with all underlying argument combinations. Lexical preferences may then be derived through constraint interaction. For example, the thought of someone eating very fast, leaving unexpressed what he eats, would turn out in English as \textit{he was eating hastily} or something similar (note that the input would certainly contain no particular lexical items, like English \textit{devour}, but a more abstract conceptual representation). The verb \textit{devour} would not be used because it is suboptimal for expressing this thought. Such a constraint-based account is essential if we want to have a uniform learning theory for syntax and the lexicon. It also bears much more explanatory potential for argument frame variation in the lexicon. Of course, this type of account presupposes the assumption of rather fine-grained constraints which are sensitive to individual lexical items—an issue which I cannot pursue any further in this book.
4.3 Candidate generation and the inviolable principles

In the representation of OT analyses, I will not use angle brackets which would suggest a subcategorization frame that is checked within the formal system of LFG. Rather I will use parantheses following the semantic form in the \texttt{pred} value of input f-structures. For example, we would get \texttt{[pred 'devour(x, y)]}. This notation is suggestive for the interpretation of dependents as semantic arguments, without the standard LFG mechanism of Completeness and Coherence being at work.

4.3.3 The base grammar $G_{\text{inviol}}$

In this subsection, an impression of the kinds of inviolable principles encoded in $G_{\text{inviol}}$ is given. As an example, assume we want to encode Grimshaw’s 1997 fragment of inversion data in English in OT-LFG, following the reconstruction by (Bresnan, 2000, sec. 2). The formal LFG-style grammar $G_{\text{inviol}}$ will have to formalize a theory of extended projections. This can be done on the basis of LFG’s extended head theory that (Bresnan, 2001b, ch. 7) discusses in detail. The principles Bresnan assumes can be fleshed out in a set of LFG rules, i.e., context-free rules\footnote{More precisely, a generalization of context-free rule notation which allows regular expressions on the right-hand side, cf. footnote 41 on page 60.} with f-annotations. Kuhn (1999b) contains a more detailed discussion of how extended head theory can be captured in concrete rule formulations; here it may suffice to assume that the effect of the principles can be envisaged as a set of classical LFG rules like the rules in (98),\footnote{DF is a generalization over discourse functions (\textsc{topic}, \textsc{focus}, \textsc{q-focus} and \textsc{subject}); CF generalizes over complement functions (\textsc{obj}, \textsc{obl4}, \textsc{comp} etc.). The non-endocentric category S that Bresnan 2000, 2001b assumes is ignored here.} generating X-bar-configurations with an extension to functional categories (like the verbal functional categories I and C). A lexical category and the corresponding functional categories on top of it form an extended projection Grimshaw (1991), Bresnan (2001b). (I do not actually assume a fixed limitation to a maximum of two functional categories, so $G_{\text{inviol}}$ really generates arbitrarily many FP’s within an extended projection.)

\begin{align*}
\text{CP} &\rightarrow (XP) (C') \\
&\quad (\uparrow \text{DF}) = \downarrow \quad \uparrow = \downarrow \\
C' &\rightarrow (C) (IP) \\
&\quad \uparrow = \downarrow \quad \uparrow = \downarrow \\
\text{IP} &\rightarrow (XP) (I') \\
&\quad (\uparrow \text{DF}) = \downarrow \quad \uparrow = \downarrow
\end{align*}
The formalization of OT Syntax in the LFG framework

\[ I' \rightarrow (I) (VP) \]
\[ \uparrow = \downarrow \quad \uparrow = \downarrow \]
\[ VP \rightarrow V' \]
\[ \uparrow = \downarrow \]
\[ V' \rightarrow (V) (XP) \]
\[ \uparrow = \downarrow \quad (\uparrow \text{cf}) = \downarrow \]

There are two crucial points to note about (98): first, the \( \uparrow = \downarrow \) annotation of both the C and the IP category in the C' rule, and the I and the VP in the I' rule. The functional head and its complement (in c-structure) act as “co-heads” on f-structure, i.e., their f-structures are identified. (All categories of an extended projection are thus mapped to the same f-structure.) Second, all categories are optional. These points together ensure that a given input f-structure has a wide range of realization alternatives—as required in an OT account with a weak \( \text{Gen}_{\text{ext}} \) component and strong effects of constraint interaction (sec. 4.2.4): since the f-structures of all heads within the extended projection are identified, each of them is a potential site for a category realizing information from the input f-structure. The structures generated for the underspecified input f-structure in (99) include the LFG analyses in (100) on page 83, for example.

\[
\begin{align*}
\text{PRED} & \quad \text{GF}_1 \quad \text{GF}_2 \\
\text{GF}_1 & \quad \text{PRED} \quad \text{GF}_2 \\
\text{GF}_2 & \quad \text{PRED} \quad \text{PRED} \\
\text{TNS} & \quad \text{FUT} \\
\text{OP} & \quad \text{Q} \\
\end{align*}
\]

The grammar specification given in (98) is not totally sufficient for deriving analyses (100b) and (100c). The f-annotation of the specifier XP in the CP rule will introduce the f-structure for what only under a discourse function, i.e. Q-FOCUS. The fact that the value of the Q-FOCUS feature and the grammatical function OBJ are structure-shared (or re-entrant) follows from an additional f-annotation not shown in (98). There are different ways of achieving the effect. Bresnan (2001b) assumes an inside-out functional uncertainty introduced by a phonologically empty object NP (or DP) under V'. I assume here an outside-in functional-uncertainty equation as an additional f-annotation in the CP rule (Kaplan and Zaenen, 1989/95, 154), shown in (101) on page 83.\(^{61}\)

\(^{61}\) The main reason for adopting the outside-in functional-uncertainty approach is presentational: the freedom of \( \text{Gen}_{\text{ext}} \), providing all kinds of candidate analyses is
4.3 Candidate generation and the inviolable principles

brought out clearer with this variant (without having to go into great technical detail). In (Kuhn, 2001c, 335) I present essentially the same fragment, but based on the inside-out functional-uncertainty specification.
The formalization of OT Syntax in the LFG framework

\[(101)\quad \text{CP} \rightarrow (XP) \quad (C') \quad (\uparrow \text{DF}) = \downarrow \quad (\uparrow \text{DF}) = (\uparrow \{\text{COMP} | \text{XCOMP}\}^* (\text{GF} - \text{COMP}))\]

where the two instances of \(\text{DF}\) are the same (i.e., both \text{TOPIC} or both \text{Q-FOCUS}, etc.) and \(\text{GF}\): the grammatical functions.

The regular expression in the feature path expression in (101) (note the operations disjunction “\(\{ \mid \}\)”, Kleene star “*” and complementation “\(\sim\)”) allows one to cover arbitrary long-distance dependencies between the topicalized argument and the verb introducing the f-structure predicate.

Many further aspects of the non-derivational framework of LFG play a role in its use as the base formalism of an OT system. I cannot go into the details here (see Bresnan (2001b)). Note however the differences between the representations in (100) and the corresponding ones in (6), sec. 2.1, which are set in the GB-based system used by Grimshaw (1997): In the c-structures in LFG, all elements are “base-generated” in their “final” position. No movements or chains are assumed; the kind of information captured by these concepts in GB-style approaches is available in LFG’s f-structures and the correspondence mapping. The OT constraints can be straightforwardly specified by referring to (c- or f-)structural configurations (see sec. 4.4).

To close this section, it should be noted that although the resulting grammar is formally an LFG grammar, it is certainly unusual since it “overgenerates” vastly, producing all universally possible c-structure-f-structure pairings. This is due to the special role that this LFG grammar plays as part of an OT model: given the different definition of grammaticality, the set of analyses generated by the LFG grammar is not the set of grammatical analyses of a particular language (as classically assumed). Rather, it is the union over all possible candidate sets—for any input.

### 4.4 The violable constraints: markedness constraints

According to the formalization in sec. 4.2, a constraint is generally a function mapping an input/candidate pair to a natural number (83). OT distinguishes two major types of constraints: faithfulness and markedness constraints. Markedness constraints are checked on the candidate analysis alone, independent of the underlying input. Faithfulness constraints are violated when the surface form diverges from the underlying form (empirical examples from phonology and syntax
4.4 The violable constraints: markedness constraints

were discussed in sec. 3.2). For this reason, the formalization apparently has to rely on a comparison of the input and the candidate analysis. It turns out that with the conception of candidate generation introduced in sec. 4.3, it is redundant to check the candidate analysis against the input; so even for faithfulness constraints, a consideration of just the candidate analysis suffices. This point will be discussed in more detail in sec. 4.5. For the time being, I will put faithfulness constraints aside and concentrate on markedness constraints. The objective is to establish a concrete way of specifying a constraint as an instance of the following restricted definition:

\[(102) \text{Restricted definition of an OT constraint}
\]

Each constraint \( C^i \in \mathcal{C} \) is a function, such that

\[ C^i (\langle T, \Phi \rangle) \in \mathbb{N}_0 \]

In sec. 4.4.1, the formulations of markedness constraints used in OT-LFG are reviewed, to provide a basis for discussion of the precise formalization. Sec. 4.4.2 addresses the universally quantified character of constraints, coming to the conclusion that this should not be reflected in the individual constraints. Sec. 4.4.3 introduces the formalization of OT constraints as constraint schemata. Finally, in sec. 4.4.4 I point out that the standard LFG means of expression are sufficient to specify OT constraints in accordance with the assumptions behind OT.

4.4.1 Markedness constraints in OT-LFG

In most OT work the constraints are formulated in prose. However, it seems to be a central assumption that the constraints can be formalized as structural descriptions of the type of representations output by \( \text{Gen}_{\text{LFG}} \)—i.e., LFG structures in our case. As a set of sample constraints, let us again look at an adaptation of the constraints that (Bresnan, 2000, sec. 2) uses in her illustrative reconstruction of Grimshaw’s 1997 fragment (parts of which I sketched in sec. 2.1). Due to their original illustrative purpose, these specific constraints do not necessarily play a role in original OT-LFG accounts (viewing OT-LFG as a theoretical paradigm, rather than just a formalism). But the means of expression used in these constraints can be regarded as representative.

The constraints involved in the derivation of \( \text{What will she read} \), which was also used for illustration in sec. 2.1, involves the constraints in (103) to be discussed in the following. The tableau for the sample candidates from (100), based on these constraints, is anticipated in (104).
The formalization of OT Syntax in the LFG framework

(103)  

a. **OP-SPEC**  
An operator must be the value of a **DF** [discourse function] in the f-structure.

b. **OB-HD**  
Every projected category has a lexically filled [extended, JK] head.

c. **DOM-HD**  
(Bresnan’s 2000 STAY (24))  
Categories dominate their extended heads.

d. **ARG-AS-CF**  
Arguments are in c-structure positions mapping to complement functions in f-structure.

(104)

```
Input:  
[  
  [PRED 'read(x, y)']  
  [G_{F_1} [PRED 'PRO'] x]  
  [G_{F_2} [PRED 'PRO'] y]  
  [TNS FUT]  
]  
```

<table>
<thead>
<tr>
<th></th>
<th>OP-SPEC</th>
<th>OB-HD</th>
<th>ARG-AS-CF</th>
<th>DOM-HD</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>[IP she will [VP read what]]*!</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b.</td>
<td>[CP what [IP she will [VP read]]]!*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c.</td>
<td>[CP what will [IP she [VP read]]]</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>

(103a) **OP-SPEC** and (103b) **OB-HD** are correspondents to Grimshaw’s constraints with the same name (cf. (1)), now based on the LFG representations. Since in $G_{invol}$ (cf. (98)) the only way of introducing something under a discourse function is as the specifier of CP or IP (and since the specifier of VP is never filled), (103a) amounts to the same as (1a) “Syntactic operators must be in specifier position”. (103b) relies on a slightly more complicated reconstruction on the placement of lexical and functional $X^0$ heads within extended projections. The concept of the extended head is defined to accommodate for the possibility of an $I^0$ or $C^0$ category to act as the head of categories further down in the same extended projection.62

---

62I made condition (ii) explicit to ensure that IP does not qualify as the extended head of $C'$ in the following configuration (cf. the discussion of (120b) below):  

(i)

```
[CP  
  [C']  
  [IP]  
]  
```
4.4 The violable constraints: markedness constraints

(105) **Definition of extended head** (Bresnan, 2001b, 132)
Given a c-structure containing nodes $\mathcal{N}, \mathcal{C}$ and c- to f-structure correspondence mapping $\phi$, $\mathcal{N}$ is an **extended head** of $\mathcal{C}$ if

1. $\mathcal{N}$ is the minimal node in $\phi^{-1}(\phi(\mathcal{C}))$ that c-commands $\mathcal{C}$ without dominating $\mathcal{C}$ [and
2. the X-bar level of $\mathcal{N}$ is less or equal to the X-bar level of $\mathcal{C}$, JK].

For our purposes, it suffices to note that there are two ways of satisfying this condition: either (i) the extended head is just the ordinary X-bar-categorial head, if present (i.e., $X^0$ or $X^0'$ for $X^0$, and $X^0$ or $XP$ for $XP$); or (ii) if there is no X-bar-categorial head, an $X^0$ category in the same extended projection (and thus mapping to the same f-structure) becomes the extended head, if it is the lowest one c-commanding the category in question ($\mathcal{C}$). This is illustrated in the following examples.

In (106a) and (106b), the categories belonging to the extended projection are circled. In (106a), all projected categories have ordinary X-bar-categorial heads: $I'$ is the head of $IP$, $I$ the head of $I'$, etc.\(^{63}\) In (106b) however, the $V$ category is c-structurally unrealized (recall that in the rules all categories are optional). So, $V'$ has no ordinary head; but the extended head definition applies, making $I$ the extended head of $V'$: it is mapped to the same f-structure and c-commands $V'$, without dominating it (and being the only such node, it is also the minimal one).\(^{64}\)

(106) a. \(\begin{array}{c}
\text{XP} \\
\text{IP} \\
\text{I'} \\
\text{VP} \\
\text{V'} \\
\end{array}\)

b. \(\begin{array}{c}
\text{XP} \\
\text{IP} \\
\text{I'} \\
\text{VP} \\
\text{V'} \\
\end{array}\)

Let us move on to constraints (103c) and (103d). For Grimshaw’s **STAY**, Bresnan also introduces a purely representational formulation, given in (103) as (103c) DOM-HD. I use an additional constraint (103d) ARG-AS-CF, to differentiate between head mobility (covered

\[\text{63}\text{The X}^0\text{ categories themselves do not have extended heads; but note that constraint (103b) OB-HD does not apply to them.}\]

\[\text{64}\text{Note that VP has an ordinary extended head again: V'}.\]
The formalization of OT Syntax in the LFG framework

by DOM-Hd) and argument mobility (covered by ARG-AS-CF). (Note that the formalism would also permit collapsing the two.\textsuperscript{65})

A further point to note about (103c) DOM-Hd is that there are two ways one may interpret this constraint, depending on how the presuppositional phrase “their extended heads” is resolved. Under the strong interpretation, DOM-Hd is violated even by categories that do not have an extended head at all, so only categories that have an extended head and dominate it satisfy the constraint. Under the weaker interpretation, categories without an extended head do not violate DOM-Hd (they do violate Ob-Hd of course). As can be seen in the tableau (104) on page 86, the interpretation adopted here is the latter one: Candidate (104b) (= (100b)) incurs no violation of DOM-Hd, although the C category has no extended head, so it does not dominate one either. A precise formulation of the interpretation adopted would be the following:

(107) DOM-Hd (revised formulation)
If a projected category has an extended head, it dominates the extended head.

Pinning down the formulation of constraints with means of a formal language, as will be done in the following, is a guarantee that ambiguities as just observed are excluded.

4.4.2 Universal quantification of constraints

To sum up the observations of the previous subsection, the violable markedness constraints in OT-LFG are formulated as conditions on the structural representations. Both f-structure and c-structure (and the relation between them) are referred to. For formalizing the primitive relations in the f-structural configurations triggering a constraint violation, a description language is already in place: the standard specification language used in LFG’s annotations, as introduced in sec. 4.1. As regards c-structure, the configurations referred to in OT constraints may

\textsuperscript{65}Bresnan’s 2000 system works with just a single \textsc{stay} constraint: (103c). The c-structural introduction of arguments in \textsc{df} positions incurs a violation of this constraint, since the inside-out functional-uncertainty approach is used (cf. the discussion of rule (101) above), so there is a phonologically empty argument \textsc{xp} under \textsc{v}, establishing identity (or structure sharing) of the f-structure under the \textsc{df} and the complement function (typically \textsc{obj}):

(i) \[
\begin{array}{c}
\text{CP} \quad \text{[DP what]} \\
\text{IP} \quad \text{she read [DP \epsilon]}
\end{array}
\]
4.4 The violable constraints: markedness constraints

go beyond the local tree accessible within a single context-free rule (the mother and the immediate daughters), but the extension is rather straightforward; it is discussed in sec. 4.4.4.

So, the problem with constraint formalization certainly does not lie in the primitive configurational relations. What is more of an issue is the overall logical structure of constraints. The constraints are not formulated with reference to a particular structure, they typically take the shape of universally quantified implications: whenever a structure satisfies the description \( A \), it should also satisfy the description \( B \) (or if the constraint is specified negatively, no structure satisfying \( A \) should also satisfy \( B \)).

A natural reaction is to try and formulate constraints as universally quantified formulae in a feature logic, to range over the complete candidate analysis to be evaluated. This move would make precise the logical structure underlying the natural language formulations of the constraints. I will pursue this idea in the following. Anticipating the result, this approach will turn out to be unnatural for modelling multiple constraint violations.\(^{66}\)

**Universal quantification**

To express universal quantification in the constraints, we need a language that permits universal quantification over the structural objects (f-structures and c-structure nodes), and that contains negation (to express implication). With a feature logic including general negation and universal quantification, we can thus express (103a) \( \text{OP-\text{SPEC}} \) as (108a).

Following B. Keller 1993, one could alternatively use a logic without the universal quantifier, but with general negation and unrestricted functional uncertainty: \(^{67}\) (108b), which is closer to the standard LFG specification language.

\[
\begin{align*}
(108) \quad \text{a.} & \quad \forall f. [\exists g. (f \text{ OP}) = g \rightarrow \exists h. (h \text{ DF}) = f] \\
\text{b.} & \quad \neg[(\exists \text{ GP}^* ) = f \land (f \text{ OP}) \land \neg(\text{DF } f)]
\end{align*}
\]

\(^{66}\)This section follows the reasoning in (Kuhn, 2001c, sec. 3.2).

\(^{67}\)Kaplan and Maxwell (1988) assumed a restricted interpretation of functional uncertainty, excluding cyclic interpretation in proving decidability of the satisfaction problem. However, Ron Kaplan (p.c., August 1999) points out that for functional uncertainty outside the scope of negation, the satisfaction problem is generally decidable (correlate of results of Blackburn and Spaan (1993)).
The formalization of OT Syntax in the LFG framework

(108b) is expressed here as an f-annotation with ↑ referring to the root node of the grammar, f is a local metavariable for an f-structure, similar to the metavariables ↑ and ↓.

For the constraints on c-structure and the c-structure/f-structure correspondence, the language has to refer to c-structure nodes and tree-geometric relations as well. The options for doing this are discussed in more detail in sec. 4.4.4. For the current purpose, these general considerations may suffice, since the approach will be rejected based on problems with multiple constraint violations.68

Constraint marking

With the general form of a constraint being a function from analyses to the natural numbers, we have yet to specify in which way the feature logic formulae are to be applied. Since they are specified in a highly general form, it is conceivable to evaluate them as if attached to the root category of the grammar (for instance, (108b) could be technically used in this way). Of course, treating the constraints as ordinary feature logic descriptions to be satisfied by the correct analysis fails to capture violability. Or in other words, the “constraints” would be a part of GenGroot.68

But there is a simple way of allowing candidates to violate the constraint formulae once per constraint: The OT constraints are disjoined with their negation, and a constraint violation mark is introduced in case their negation is satisfied. Assume constraint C1 is specified by the feature logic formula ψ1, then we can model its application as a violable constraint as

\[ \psi_1 \lor (\neg \psi_1 \land \exists^* C1 \in (\uparrow \text{MARKS})) \]

attached to the root node. From this MARKS set, the constraint marks can be simply read off: 1 if the constraint mark is in the set; 0 otherwise.69

68 Note that decidability is not an issue with such a logic: Although the general satisfiability problem for feature logics with universal quantification and general negation is undecidable (B. Keller 1993, sec.4.4, Blackburn and Spaan, 1993, sec. 5), the use of this logic for constraint checking on given candidate analyses is unproblematic, since the expressions are not used constructively. (This was pointed out by Ron Kaplan (p.c.) and Maarten de Rijke (p.c.).) The task performed is not checking satisfiability, but model checking, which is easy: the given candidate structure has a finite number of nodes and f-structures, thus it can be checked for each of them whether it satisfies the constraints by instantiating the variables to the given elements.

69 Note that the result will be an LFG grammar that models not only the function
4.4 The violable constraints: markedness constraints

Multiple constraint violations

What happens in a scheme based on the mechanism (109) when a candidate analysis contains several instances of the configuration excluded by the constraint? Obviously, the formula $\psi^1$ cannot be satisfied, so the other disjunct is picked, introducing a single violation mark. Of course, multiple violations of a given constraint up to a fixed upper bound could be simulated by formulating extra constraints that check for the presence of several instances of the violation in the candidate structure.\textsuperscript{70} This may be acceptable when the domain of competition is locally confined to non-recursive structures, but it is unnatural for the fully recursive generative system of syntax.

If the possibility of multiple constraint violations is to be granted generally and without an upper bound, the mechanism checking for constraint satisfaction has to be modified. As the candidate structure is traversed, the constraint has to be checked over and over. Whenever the application of a constraint leads to inconsistency, a violation has to be counted, but the rest of the structure has to be checked for further violations of the same constraint.

Since the constraint checking has to traverse the candidate structure anyway, one may ask if there is still the need for formulating the constraint in a universally quantified way. Note that the original idea of this format was to ensure that the constraint ranges over the entire candidate structure (and not just the outermost f-structure, to give a concrete example). Moreover, is it clear at all for arbitrary constraints in such a highly expressive logic what constitutes a multiple violation? For simple implicational constraints with universal quantification over one structural element (an f-structure or a c-structure node) it is intuitively clear what it means to violate this constraint more than once; but it seems that expressions involving more than one universal are more problematic. Assume we wanted to work with the following constraint (110a):

\begin{itemize}
  \item \textit{Gen}_{\text{when used to generate from an input f-structure}, \text{but also the function marks.}}
  \item We may thus call the grammar \textit{G}_{\text{input, marks}}. I will briefly come back to this idea in sec. 4.4.5.
  \item Karttunen (1998) proposes this for his computational model of OT phonology, which does not allow arbitrarily many violations of a constraint either. For certain constraints, multiple violability can however be modelled with finite-state means, as pointed out by Gerdemann and van Noord (2000) (see Jäger (2002b) for a generalization).
\end{itemize}
The formalization of OT Syntax in the LFG framework

(110) Hypothetical constraint

a. For all DP categories, all their daughters are nominal (i.e., either N or D projections).

b. \[ \forall n. [\text{DP}(n) \rightarrow \forall m. [\text{dtr}(m, n) \rightarrow \text{nom}(m)]] \]

Now, the following structures are evaluated:

(111) a. VP

```
  VP
   |
  DP   DP
     |
    D  VP, NP
```

b. VP

```
  VP
   |
  DP   DP
     |
    AP  VP, NP
```

c. VP

```
  VP
   |
  DP   DP
     |
    D  VP, AP
```

None of the three satisfies (110). But how many violations does each of them incur? In (111a) and (111b), one DP fails to satisfy the condition that all its daughters are nominal, while in (111c), both do. So, under one interpretation, (111a) and (111b) should violate (110) once, and (111c) twice.

On the other hand, (111a) is better than (111b), because only one of the DP’s daughters violates the inner implication. Shouldn’t one expect then that (111b) incurs two violations? In fact, the modified checking mechanism sketched above, which counts sources of inconsistency, would presumably have this effect (unless the mechanism is explicitly set back to the outermost universal whenever an inconsistency is encountered).

The problem is also clearly brought out if we look at the following two reformulations of (110):

(112) a. Each daughter of a DP category is nominal.

\[ \forall m. [\exists n. [\text{dtr}(m, n) \land \text{DP}(n)] \rightarrow \text{nom}(m)] \]

(113) a. For every category, if it has a non-nominal daughter, then it is not a DP.

\[ \forall n. [\exists m. [\text{dtr}(m, n) \land \neg \text{nom}(m)] \rightarrow \neg \text{DP}(n)] \]

Both are equivalent to (110) in terms of classical predicate logic, but read with the intuitions behind violable constraints, they clearly differ
4.4 The violable constraints: markedness constraints

in the number of violations ascribed to (111b): (111b) violates (112) twice, but it violates (113) only once. This indicates that the use of general formulae of this type of feature logic is inappropriate for modelling the intuitions behind OT constraints, for which we need a precise way of stating what it means to incur multiple violations of a given constraint.

4.4.3 Constraint schemata

The recursive applicability of constraints over the entire candidate structures has to be inherent to the general constraint marking mechanism in order to allow arbitrary multiple violations. Hence, the universal range does not have to be stated explicitly in each individual constraint. To the contrary, this makes the constraints counterintuitive, as was discussed in the previous section.

Thus, we should give the individual constraint formulations a simpler logical structure—this is also in line with the methodological principles discussed particularly in sec. 2.2.4 (cf. also Grimshaw (1998)). Now, the universal applicability is implicit to all constraints and will be made effective in the checking routine that the candidate structures have to undergo after they have been constructed. At every structural object (either a c-structure node or an f-structure), all constraints are applied. This application of the constraints to multiple objects is the only source for multiple violations—a single structural element can violate each constraint only once. At each application, the individual constraints are again interpreted classically.

In order for this to work, the structural object which is being checked with a given constraint has to be clearly identified. I will assume a metavariable ∗ for this (reminiscent of the ∗ used in standard LFG f-annotations of categories to refer to the category itself, cf. page 61). (112) will for example take the following shape:

\[(114) \exists n. [dtr(\star, n) \land DP(n)] \rightarrow nom(\star)\]

When the constraints are checked, the metavariable ∗ will be instantiated to one structural element after the other. Thus, the constraints are actually specified as constraint schemata, generating classical constraints.

Note that we could now express (110b) in either of the following two ways, reaching the two different effects for the structures in (111) discussed above:
The formalization of OT Syntax in the LFG framework

(115) a. DP(*) → ∀m.[dtr(m, *) → nom(m)]
   b. ∀n.[DP(n) → (dtr(*, n) → nom(*))]

Expressing (115b) in the equivalent form (114) may actually be more intuitive (note that now, equivalences of classical logic apply again).

So, we can state the following restriction on constraint formulation, which allows for a simple concept of multiple violations that is compatible with a classical concept of satisfiability, and also meets the intuitions behind violable constraints in OT:

(116) Restriction on the form of constraints
       Violable constraints are formulated with reference to a unique structural element, which is referred to by a metavariable (*).

Scalar constraints

Note that it is compatible with this restriction to assume a “scalar” interpretation of alignment constraints like, e.g., HEAD LEFT (117) from Grimshaw (1997).71 (Under a scalar interpretation, this constraint is violated twice if there are two intervening elements between a (single) head and the left periphery of its projection.)

(117) HEAD LEFT: (Grimshaw, 1997, 374)
       The head is leftmost in its projection.

The metavariable-based formulation allows a clear distinction between the non-scalar and the scalar version of this constraint as shown in (118). (The function proj(n) is assumed to denote the projection of the node n; the relation dtr(n, m) holds if n is a daughter of m, the relation precede is obvious.)

(118) HEAD LEFT
       \[
       \begin{align*}
       \text{non-scalar interpretation} & : \text{head}(*) \rightarrow \neg \exists n. [\text{dtr}(n, \text{proj}(*)) \land \text{precede}(n, *)] \\
       \text{scalar interpretation} & : \text{cat}(*) \rightarrow \neg \exists n. [\text{head}(n) \land \text{dtr}(*, \text{proj}(n)) \land \text{precede}(*, n)]
       \end{align*}
       \]

The first formulation is stated from the point of view of the head; since the instantiated schema is interpreted classically (i.e., incurring maximally one constraint violation for each structural element), a given

71In recent work, Sells (e.g., 1999, 2001) has proposed an antisymmetric constraint system for deriving the typologically attested space of c-structure configurations, using alignment constraints of this kind.
4.4 The violable constraints: markedness constraints

head can violate this constraint only once (even if there are several intervening nodes to the left of it). The second formulation is from the point of view of the intervening category; thus if there are several of them, the overall structure will incur several violations of this constraint.

Formalization of the example constraints

With the formal means of constraint schemata, we are in a position to give the example constraints from (103) a precise formulation. English paraphrases for the formal specifications are given below the formulae. It is assumed that all operators introduce a feature $\text{OP}$ (question operators have $\text{OP}$-value $Q$, for example), non-operators do not introduce this feature. $f\text{-str}(f)$ holds of f-structures, $\text{cat}(n)$ of c-structure nodes.

(119) a. **OP-SPEC** Bresnan (2000)
An operator must be the value of a DF in the f-structure.

\[(f\text{-str}(\star) \land \exists v. [([\star \text{OP}] = v)])
\rightarrow \exists f. [f \text{DF} = \star] \]

“If $\star$ is an f-structure bearing a feature $\text{OP}$ (with some value), then there is some (other) f-structure $f$ such that $\star$ is embedded in $f$ under the feature $\text{DF}$.”

b. **OB-HD** (Bresnan, 2000, (21))
Every projected category has a lexically filled [extended, JK] head.

\[(\text{cat}(\star) \land (\text{bar-level}(\star, 1) \lor \text{bar-level}(\star, 2))) \rightarrow \exists n. [\text{ext-hd}(n, \star)] \]

“If $\star$ is an X-bar or X-max category, then there is some node $n$ which is the extended head of $\star$.”

c. **DOM-HD** (revised formulation)\(^{72}\)
If a projected category has an extended head, it dominates the extended head.

\[\forall n. [(\text{cat}(\star) \land \text{ext-hd}(n, \star)) \rightarrow \text{dom}(\star, n)] \]

“For all nodes $n$ such that category $\star$ is their extended head, $n$ dominates $\star$.”

\(^{72}\)Note that the alternative, stronger interpretation of STAY discussed in connection with (107) can be easily expressed too:

\[(\text{cat}(\star) \rightarrow (\exists n. [\text{ext-hd}(n, \star) \land \text{dom}(\star, n)])])\]
The formalization of OT Syntax in the LFG framework

d. **ARG-AS-CF**
Argument are in c-structure positions mapping to complement functions in f-structure.

\[ \exists f.([f \ CF] = \star) \rightarrow \exists n. [\text{cat}(n) \wedge \phi(n) = \star \wedge \text{lex-cat}(\mathcal{M}(n))] \]

"If \( \star \) is embedded under a complement function \( CF \) (in some \( f \)-structure \( f \)), then there exists a c-structure node \( n \) projecting to \( \star \), whose mother is a lexical category, i.e., \( \star \) is c-structurally introduced as the complement of a lexical category (the canonical position for \( CF \) introduction, according to the mapping principles, (Bresnan, 2001b, sec. 6.2))."

When these constraints are applied to the sample candidate set of (100)/(104), we get the following constraint violations listed under the analyses:

(120) **Candidate analyses for tableau (104), with \( \phi \) mapping shown in selected cases**

a. \[
\begin{array}{c}
\text{IP} \\
\text{NP} \\
\text{she} \\
\text{will} \\
\text{VP} \\
\text{read} \\
\text{NP} \\
\text{what}
\end{array}
\]

\*OP-SPEC

b. \[
\begin{array}{c}
\text{CP} \\
\text{NP} \\
\text{what} \\
\text{IP} \\
\text{she} \\
\text{will} \\
\text{NP} \\
\text{read}
\end{array}
\]

\*OB-HD, \*ARG-AS-CF
4.4 The violable constraints: markedness constraints

The constraint violations are derived as follows: In candidate (120a), we have an f-structure that satisfies the antecedent of the implication (119a), the Op-Spec constraint: the f-structure under Obj bears the feature OP Q. Let us call this f-structure ★ for the moment. To satisfy Op-Spec, ★ has to be embedded in some f-structure under a feature DF. This is not the case in candidate (120a), thus we get a violation ★Op-Spec. Note that both of the two other candidates satisfy Op-Spec, since there the OBJ value is identical with the value of Q-Focus (an instance of DF).

The f-structure ★ we were looking at in candidate (120a) does however satisfy the remaining three constraints: Constraints (119b) and (119c) are satisfied trivially, since our ★ is not a c-structure category so the antecedent is already false (making the implication true). However let us look at constraint (119d) Arg-as-CF: with f instantiated to the outermost f-structure (the only possibility), the antecedent is satisfied: ★ is indeed embedded under a CF, namely OBJ. So does ★ also meet the consequent? There is only one category projecting to ★: the lower NP node, and fortunately its mother—V'—is a lexical category, as required. So we get no violation of Arg-as-CF. All other c-structure and f-structure elements of candidate (120a) satisfy all four constraints, so ★Op-Spec is the only constraints violation we get.

Let us now look at candidate (120b). Since there is no C category in the tree and IP does not qualify as the extended head of C' (cf. definition (105) and footnote 62 on page 86), we get a violation of (119b) Ob-HD. Checking (119d) Arg-as-CF, we can again instantiate f as the outermost f-structure, and ★ as the f-structure under OBJ. There is a single category node mapping to ★: the NP in the specifier to CP.
(note that the f-structure under OBJ and Q-FOCUS is identical). In order to satisfy the constraint, the mother node of this NP—CP—would have to be a lexical category. This is not the case, so we get *ARG-AS-CF. The other constraints are fully satisfied.

Candidate (120c) shares with (120b) the configuration violating ARG-AS-CF. If we look at C' however, we note that (119b) O8-HD is satisfied here: since C is filled we do find an extended head. Note that I' too has an extended head, although there is no X-bar-categorial head I; C is in the appropriate c-commanding position and has the right bar-level. However, I' violates (119c) DOM-HD: since it has an extended head, it would also have to dominate this head for satisfying DOM-HD.

With the constraint ranking OP-PEC, O8-HD ∋ ARG-AS-CF, DOM-HD for English, we get (120c) as the most harmonic of these three (and in fact all possible) candidates, as shown in tableau (104) already.

Constraint marking

Based on the constraint schemata proposed in this section, the marking of constraint violations incurred by a given candidate structure is conceptually straightforward. For each structural element (c-structure node and f-structure), the set of constraints is applied, with the metavariable instantiated to the respective element. When the application of a constraint fails, the candidate is not rejected, but the count of constraint violations is increased.

So, the constraint marking function from candidate analyses to natural numbers can be given a precise specification, based on the cardinality of the set of structural elements for which the instantiated constraint schema is not satisfied by the candidate analysis: 73

\[
C^i((T, \Phi)) = \{ x \mid x \text{ is a structural element of } T, \Phi \text{ and } (T, \Phi) \not\models \psi^i[*/x] \},
\]

where \( \psi^i[*/x] \) is a constraint schema \( \psi^i \) formalizing \( C^i \), with \( x \) instantiating the metavariable \( * \).

4.4.4 Constraint schemata as standard LFG descriptions

In this section, I adjust the formulation of OT constraints further, based on the observation that it can be brought closer to the formulation of

73The adoption of a set-theoretical view was suggested by Ron Kaplan (p.c.) and Dick Crouch (p.c.).
4.4 The violable constraints: markedness constraints

familiar, classical constraints in the LFG framework. The expressiveness is slightly reduced by this move, which should be considered an improvement, since the linguistically relevant constraints for OT syntax can still be expressed.

OT constraints of c-structure

The constraint formalization of the previous section (and Kuhn (2001c)) makes use of intuitively named predicates over categories and tree configurations (e.g., \textit{ext-hd, dom, lex-cat}, etc.). A precise definition for these predicates was not given since the focus of presentation was on the question how to formalize the idea of multiply violable constraints. Nevertheless we need to find a more explicit account.

A slight difficulty in this task is that the intuitive difference between classical LFG constraints and typical OT constraints seems larger for constraints on c-structure than for constraints on f-structure. In the latter case, propositions about certain graph configurations have classically been expressed as Boolean combinations of various types of equations. This can easily be transferred to violable OT constraints, which add certain implications etc. For constraints on c-structure, the classical means of expression in LFG have been context-free rewrite rules (extended to allow regular operations such as optionality, union/disjunction, and Kleene closure). Here, it is not so obvious how the typical implications of OT constraints should be added. For instance, at what level should the difference between the \texttt{I'} nodes in the following tree structures (one satisfying, one violating \texttt{Ob-Hd}) be stated?

(122) a. CP
   \hspace{1em} NP \hspace{1em} C' \hspace{1em} IP \hspace{1em} NP \hspace{1em} I' \hspace{1em} VP

   b. CP
   \hspace{1em} NP \hspace{1em} C' \hspace{1em} IP \hspace{1em} NP \hspace{1em} I' \hspace{1em} VP

There are various possible formats for stating such c-structure constraints. Underlying the previous section was a tendency to move away from the standard means of grammar specification in LFG, suggesting the introduction of a totally new tree specification language. So, on the one hand the trees in (122) are constructed according to the inviolable
The formalization of OT Syntax in the LFG framework

constraints in Gen\textsubscript{G-slot}, which are formulated in the form of (extended) context-free rewrite rules. But for the violable OT constraints, a new, more general tree description language has been assumed.

For an analysis of further computational properties of the account, this move is problematic, since it would make the transfer of formal results from classical LFG to our situation unnecessarily difficult. It is also possible to state the constraints in a more familiar format.

We have still at least two choices: one could follow the proposal of Andrews and Manning (1999), who modify the classical projection architecture of LFG to encode categorial structure as attribute-value matrices (inspired by HPSG\textsuperscript{74}). In such a framework, implicational OT constraints could be expressed as combinations of feature equations describing these attribute-value matrices.

The other alternative, which I will follow here, does not involve importing c-structure information into the feature structure representation. The additional constraints on c-structure are expressed as regular expressions over sequences of categories in local subtree configurations.\textsuperscript{75} For instance, a constraint could demand that in a local subtree dominated by I', an I daughter is present: (123).\textsuperscript{76} The general format of constraints is implicational \( A \Rightarrow B \), to be read as “if a local configuration satisfies description \( A \), then it also has to satisfy description \( B \) in order to satisfy the constraint”.

(123) **Illustrative constraint:** I'-HAS-I-HEAD

\[ *=I' \Rightarrow \ * \]

\[ ?* I ?* \]

With just the ordinary set of c-structure symbols available, this account has a very limited expressiveness. The different status of (122a) and (122b) could not be detected, since at the local subtree level, the two representations are indistinguishable.

However, we can extend the set of category symbols to allow for more fine-grained distinctions to be made. The standard convention

\textsuperscript{74}Head-Driven Phrase Structure Grammar, Pollard and Sag (1994)

\textsuperscript{75}I use the term local subtree to refer to subgraphs of trees with depth 1.

\textsuperscript{76}As in the XLE system’s rule notation, the '?' is assumed to denote arbitrary category symbols; the stars ('*') are Kleene stars. So, the regular expression on the lower side of the local tree in (123) is matched by all category strings containing at least one occurrence of I.
4.4 The violable constraints: markedness constraints

is to encode the lexical class (N, V, A, P), the bar level and the functional/lexical distinction in the category symbols. Some of this information would be recoverable from the tree structure; for example, if the lexical class was only encoded in the \(X^0\) categories and not in their projections, one could still detect it by tracing down the X-bar projection line.\(^77\) With the explicit encoding of lexical class in higher projections, this information is made available up to the XP level, where it is required to express constraints on possible combinations of maximal categories with other material. So, the definition of the X-bar categorical head is effectively precomputed: the rewrite rules are specified in a way that ensures that the lexical class of a projection is the same as the one of its projecting \(X^0\) head.

Now, the same idea of precomputing relevant information from the c-structure configuration and providing it as part of a category name can be applied in other situations. For instance we could distinguish between IP ccxhy and IP ccxhn (for c-commanding extended head yes/no), based on rules like the following:

\[
\begin{align*}
(124) & \quad C' & \to & C \text{ IP ccxhy} \\
& \quad C' & \to & \text{ IP ccxhn} \\
& \quad \text{IP ccxhy} & \to & (XP) \text{ I' ccxhy} \\
& \quad \text{IP ccxhn} & \to & (XP) \text{ I' ccxhn} \\
& \quad \text{I' ccxhn} & \to & (I) (VP) \\
& \quad \text{I' ccxhy} & \to & (I) (VP)
\end{align*}
\]

The symbol IP ccxhn is introduced in the second of the two \(C'\) productions, i.e., when a c-commanding extended head (i.e., the \(C\) category) is missing: this information is propagated down to the I’ level. (Of course, the same construction would have to be introduced for the VP, but this is omitted here for simplicity.) We can now formulate the exact conditions for the \(OB-HD\) constraint at this level:

\[
(125) \quad \text{Partial specification of constraint: } OB-HD \quad \star = \text{I' ccxhn} \Rightarrow \star \quad \frac{\star}{\star} \quad \frac{\star}{\star}
\]

Without the c-commanding extended head being present, the I’ has to

\(^77\)This reasoning is based on a more classical set-up, where the \(X^0\) heads cannot be missing; to extend it to the current LFG set-up where all categories including heads are optional, one might assume that projections with missing \(X^0\) heads are underspecified for lexical class.
The formalization of OT Syntax in the LFG framework

have a local I head in order to satisfy OB-HD; when a c-commanding
extended head is present, there are no restrictions on the local subtree.

With the described technique of threading the relevant information
through the c-structure skeleton, all distinctions with a finite set of
relevant choices can be “imported” to the scope of a local subtree.

Explicit generalizations over c-structure

A possible objection to the proposed scheme of formulating c-structural
constraints is that generalizations are not expressed explicitly in the
Gen_{Ob-Hd} rules and in the OT constraints. Instead, a massively disjunc-
tive specification seems to be required for non-trivial systems.

However, there are several ways of ensuring that at the level
of grammar specification, the underlying generalizations are indeed explicit. In (Kuhn, 1999b, 4.1), I distinguish two strategies:
the representation-based vs. the description-based formulation of c-
structural generalizations. In both cases, the idea is that entire classes
of categories can be constrained with a single rule or principle. The
representation-based strategy modifies the representation of categ-
ories, no longer viewing them as atomic symbols but assuming an
internal structure (thereby making explicit a conceptual classification
that is assumed anyway). The IP category, for instance, can be seen as
a combination of category-level “features” for lexical class \( V \), bar-level
2 (or maximal), and category type functional, which we may write as
\( \langle V, 2, \text{func} \rangle \). It is obvious how further distinctions can be added to this
scheme (to give us \( \langle V, 2, \text{func}, \text{ccxhy} \rangle \), for instance, to signal the exis-
tence of a c-commanding extended head). As long as all features have
a finite range of values and there is a finite number of features, the
fundamental LFG set-up remains unchanged. But now, generaliza-
tions over rules can be expressed more explicitly by constraining the
c-co-occurrence of category features within rules. In particular, OT con-
straints can be formulated that express implications based on particular
category features, leaving the rest underspecified. A more general
specification of (125) would be

\[
\text{(126) Generalized (partial) specification of constraint: OB-HD} \\
* = \langle \alpha, 1 \lor 2, \beta, \text{ccxhn} \rangle \Rightarrow * \triangleleft \langle \alpha, 0, \beta, \_ \rangle ?^* 
\]

The XLE implementation provides the discussed concept of complex categories, con-
straints about which are formulated in parametrized rules. For the category-level features
a positional encoding is used; the format for complex categories is \( \text{XP} [\alpha, \beta] \).
4.4 The violable constraints: markedness constraints

Note that as a prerequisite for this technique of constraint specification, principles about the inheritance of all relevant category-level features in the $G_{viol}$ have to be explicitly stated. It is beyond the scope of the present discussion to go into details of such a specification. I will assume that $G_{viol}$ is specified in such a way that (i) the c-structure in all candidate analyses is in accordance with the intuitive definition of the properties encoded by the category-level features, and (ii) all combinations of categories conforming with the definitions are effectively generated. The $G_{viol}$ grammar specification is simply taken as the formal definition of all category-level features involved, so there is no danger of generating any incorrect candidates (in a technical sense).

The other strategy for expressing generalizations about c-structure (the description-based formulation) keeps up atomic category symbols, but provides a richer language for describing categories: the category IP is in the denotation of meta-category names like V-cat, Xmax-cat, and Func-cat, which can be intersected as appropriate. This strategy can be directly integrated into the classical regular-language specification of LFG rules. However, it does not allow one to generalize over entire classes of meta-categories, so I keep to the representation-based strategy assuming complex categories.

Excluding universal quantification from the constraint schemata

Having simplified the formulation of c-structural OT constraints, we should now take another look at universal quantification (which is not included in the means of expression of standard LFG). According to sec. 4.4.3 (and Kuhn (2001c)) universal quantification is still allowed in the constraint schemata, although it is no longer used to account for multiple constraint violations (cf. the discussion of sec. 4.4.2). In the formulation of sample constraints in sec. 4.4.3, we do find a use of the universal quantifier, in the DOM-Hd constraint (119c), repeated below.

(119) c. DOM-Hd (revised formulation)

If a projected category has an extended head, it dominates the extended head.

\[
\forall n. [(\text{cat}(\star) \land \text{ext-hd}(n, \star)) \rightarrow \text{dom}(\star, n)]
\]

“For all nodes $n$ such that category $\star$ is their extended head, $n$ dominates $\star$.”

Clearly, this universal quantification has been made obsolete by the move to the more canonical formulation of c-structural OT constraints,
The formalization of OT Syntax in the LFG framework

according to which the property of having an extended head is now encoded in the c-structure categories. But even without this move, the use in (119c) would not justify the need for full universal quantification: according to the definition of extended head (see (105), adapted from (Bresnan, 2001b, 132), and note the minimality condition), the relation \( \text{ext-hd}(A, B) \) could be rewritten in a functional notation, with \( \text{ext-hd}'(B) = A \). So, (119c) could be reformulated as follows:\(^79\)

\[
(127) \quad \text{cat}(\star) \rightarrow \text{dom}(\star, \text{ext-hd}'(\star))
\]

Further uses of universal quantification occur in the two versions of the hypothetical constraint first introduced in (110) and discussed in sec. 4.4.3. Their schema-based specification is repeated below.

\[
(115) \quad \begin{align*}
\text{a.} & \quad \text{DP(\star)} \rightarrow \forall m. [\text{dtr}(m, \star) \rightarrow \text{nom}(m)] \\
\text{b.} & \quad \forall n. [\text{DP}(n) \rightarrow (\text{dtr}(\star, n) \rightarrow \text{nom}(\star))]
\end{align*}
\]

Concerning (115b), we can again exploit the functional character of the daughter relation: \( \text{dtr}(A, B) \) iff \( M(A) = B \), which gives us

\[
(128) \quad \text{DP}(M(\star)) \rightarrow \text{nom}(\star)
\]

In (115a), we do have a “real” example of universal quantification. The intuition is that in order to satisfy this constraint, all daughters of category \( \star \) have to be nominal. But since the property checked for in the constraint—having nominal daughters exclusively—is locally confined, it would be possible to introduce a complex-category-level distinction on mother categories encoding whether or not they have this property. Then the constraint would only need to check for the relevant category feature. Alternatively, a binary-branching re-coding of the c-structure could be assumed and the property \( \text{nom} \) could be checked with a simple constraint schema at each level, so the universal-quantification effect would be shifted to the universal application of constraint schemata.

I take these examples to indicate that after the move to constraint schemata, which are evaluated at every structural object, universal quantification is no longer required for linguistically interesting constraints. In fact it goes against the general principle of keeping the individual constraints simple and exploiting constraint interaction for higher-level effects.

\(^79\) \text{ext-hd}' \text{ is a partial function—some categories have no extended heads. One might define a proposition containing an undefined functional expression to be false, then the consequent of (127) would become false, and the entire constraint would become true/satisfied for a category which has no extended head.}
4.4 The violable constraints: markedness constraints

This means in particular that for the f-structural OT constraints we can restrict the format available for the formulation of constraint schemata to the format of standard LFG f-annotations (with the metavariable $\star$ added). Note that this excludes constraints like the following, universally quantifying over all possible feature paths in an f-structure:

\begin{equation}
\forall \text{PATH}, f.[(\star \text{ PATH}) = f \rightarrow (f \text{ CHECK}) = +]
\end{equation}

This constraint checks an entire f-structure with all substructures for a particular property ([CHECK $+$]). Note that when this schema is instantiated with different partial f-structures, we get a funny behaviour: a violation originating from a certain embedded f-structure $g$ lacking the specification [CHECK $+$] is counted over and over again; so, we get multiple violations of the constraint, but the number of violations does not depend on the number of offensive partial f-structures, but rather on the size of the overall f-structure. A more reasonable behaviour results if we use the metavariable for the local f-structure that is demanded to have the specification [CHECK $+$]. In other words, we do away with universal quantification within the constraint schemata.

So, concluding this section, the constraints in the OT-LFG setting can be restricted to (i) conditions on local subtrees in c-structure (making use of complex category specification for generality), and (ii) functional annotations containing the metavariable $\star$. In addition, we allow for a combination of c-structural and f-structural restrictions, so OT constraints can express conditions on the structure-function mapping. This restricted format will be most relevant for decidability considerations discussed in sec. 6.2. For other discussions in the following, I will keep up the more perspicuous constraint schema format using intuitive predicates and universal quantification.

4.4.5 Digression: Constraint marking as description by analysis vs. codescription

Given the schema-based definition of constraints, the overall constraint violation profile of a given candidate results when the counts for the individual constraints are taken together. The way candidate set generation and constraint marking have been defined suggests a clear conceptual split between the two abstract processes: Candidate generation is prior to constraint marking, and one may want to think of the latter starting only once the former has finished. However, with the restriction to constraint schemata referring to single structural elements, an
The formalization of OT Syntax in the LFG framework

alternative way of conceiving of constraint marking is opened up: we may envisage candidate generation and constraint marking as a combined abstract process. This alternative view has certain advantages, for instance in practical grammar writing, where the constraint marking can be coded into the grammar rules specifying the inviolable principles. As long as the effect of the constraint schemata is spelled out for every single element in the grammar, there is no difference in the resulting OT system, however coding the marking into the grammar makes it possible to be selective (which has certain advantages, but bears some risks too). In this section, I will make a few remarks about the two views.

There is a parallel to different approaches of realizing syntax-driven semantic analysis in an LFG grammar: the description by analysis approach vs. the codescription approach. In the former, semantic construction starts only once the f-structure analysis has been created; in the latter, f-structure and semantic structure are built up simultaneously (see e.g. Kaplan (1995) for discussion). The constraint marking approach working with two separate abstract processes works as description by analysis. The constraint violation profile of a candidate (corresponding to semantic structure) is built by traversing the previously constructed syntactic analysis.

But as long as there is a unique place in the grammar/lexicon from which a certain structural element can be introduced, we can attach the constraints it has to meet in the rule or lexicon entry already. This becomes possible since the constraints are formally restricted to refer to a single structural element. What we have got now is a codescription approach. For c-structure categories, it is true that there is such a unique place: we can attach the constraints in the rules for nonterminals and in the lexicon entries for terminals. With f-structures, we have to be careful, since due to unification, there is not generally a unique source in the grammar/lexicon for an f-structure. For all PRED-bearing f-structures, the instantiated symbol interpretation of semantic forms (cf. page 64) guarantees uniqueness however. Hence, attaching the constraints wherever a PRED-value is introduced captures this subset of f-structures, and a generalization to all f-structures is possible.\footnote{The generalization would work as follows: even for f-structures without a PRED-value, a feature taking an instantiated symbol as its value is assumed. Let us call it ID. The Completeness condition is extended to demand that in a well-formed f-structure, all substructures contain this feature ID. The value for ID is introduced in the constraint marking schemata for f-structures, which are optionaly applied in all rules and all lexicon entries, whenever there is a reference to an f-structure. Since the value is an instantiated...
4.4 The violable constraints: markedness constraints

This means that a general conversion from description by analysis to codescription—and vice versa—is possible.

As in the grammar-based constraint marking discussed briefly in sec. 4.4.2 (cf. footnote 69 on page 91), we have to ensure the violability of the attached OT constraints. We can again disjoin the constraints with their negation—this time not just for the root category, but potentially for every category in each rule. As a result we will again have a grammar that performs both the task of candidate generation and constraint marking; thus we might call it $G_{enviol, marks}$.

Constraint violation marks are introduced to the MARKS multiset in the places in the grammar where the structure violating a constraint is created. For example, we may want to formulate the constraint (119c) DOM-HD by making the following annotation in each of the grammar rules (cf. (98)):

\[
(130) \quad C' \rightarrow \left\{ \begin{array}{c} \downarrow C \\ \uparrow = \downarrow \end{array} \right. \quad \text{DOM-HD}^{\epsilon} \in (\uparrow \text{MARKS}) \quad \text{(IP)}
\]

In the original rule, the head C was simply optional “(C)”, now we have an explicit disjunction of C and the empty string $\epsilon$. At c-structure, this is equivalent ($\epsilon$ is not a phonologically empty category, it is indeed the empty string in the formal language sense). However, at f-structure, we can now annotate the option not realizing C with the introduction of the constraint violation mark $\ast \text{DOM-HD}$.

Since constraint marks can now be introduced into the MARKS multiset in every rule, we have to ensure that all contributions are collected and made available at the root node of the analysis.\footnote{As Ron Kaplan (p.c.) pointed out, the collection of marks need not be realized within the grammar of course, since evaluation is a grammar-external process anyway; thus, the identification of the MARKS feature is an unnecessary overhead.} This can be achieved by identifying the MARKS feature of all daughter constituents with the mother’s by the equation $(\uparrow \text{MARKS}) = (\downarrow \text{MARKS})$, creating a single multiset for the complete analysis. Note that multiple violations of a single constraint fall out from the use of a multiset.\footnote{An alternative way using a standard set would be to interpret the constraint marks symbol, only one application of the schemata can be performed on each f-structure. When there is unification, i.e., other rules or lexicon entries referring to the same f-structure, the optionality of application will ensure that they are not applied again. But Completeness ensures that they are applied once. Of course there is a non-determinism leaving open which rule/lexicon entry is the one setting the ID value. But this does not affect the result.}
The formalization of OT Syntax in the LFG framework

If we now use a special projection $o$ instead of the feature Marks (so the membership constraint from (130) reads as “$\text{DOM-Hd} \in o$”) we are very close the system of Frank et al. (2001), which is built into the Xerox Linguistic Environment (XLE) LFG parsing/generation system. The projection $o$ is always a multiset of constraint violation marks and we can assume implicit trivial equations ($\uparrow = \downarrow$) in the rules, so the $o$-structure of all constituents is identified. The XLE system also provides an (extended) implementation of the Eval-function, based on the marks introduced to the $o$-projection, and a dominance hierarchy specified in the configuration section of the grammar.\footnote{As Frank et al. (2001) discuss in detail, XLE distinguishes several types of constraint marks—in particular preference marks besides dispreference marks. For the purposes of modelling a standard OT account, the dispreference marks suffice.}

One advantage of the grammar-based or codescription approach is that often the constraint formulation becomes simpler: the triggering configurations for constraint satisfaction/violation do not have to be restated when they have an obvious location in the grammar specification. A good example is constraint (119d) ARG-AS-CF:

\begin{equation}
\text{(131) ARG-AS-CF} \tag{131}
\end{equation}

Arguments are in c-structure positions mapping to complement functions in f-structure.

**Description-by-analysis formulation**

\[ \exists f.[(f \text{ CF}) = \star] \rightarrow \exists n.[\text{cat}(n) \land \phi(n) = \star \land \text{lex-cat} (\mathcal{M}(n))] \]

**Codescription formulation** (just adding the violation mark introduction to rule (101))

\[
\begin{array}{ccc}
\text{CP} & \rightarrow & (\text{XP}) \\
(\uparrow \text{DF}) &=& \downarrow \\
(\uparrow \text{DF}) &=& (\uparrow \{\text{COMP} | \text{XCOMP}\} \ast (\text{GF-\text{COMP}})) \\
\ast \text{ARG-AS-CF} \in o \\
\end{array}
\]

The description-by-analysis approach with the strict separation of candidate generation and constraint marking makes it necessary to reanalyze the c-structure configuration in which complements of lexical categories are mapped to the embedded CF. In the codescription account, a constraint violation mark can be introduced when an argument is c-structurally introduced in the non-canonical position, making use of functional uncertainty.\footnote{The two formulations are not strictly equivalent. When an argument is realized in

\begin{itemize}
\item introduced as instantiated symbols (like the \text{PRE}\text{D} values in standard LFG), i.e., as pairwise distinct.
\end{itemize}
4.5 Faithfulness constraints

A further, practical advantage of the grammar-based constraint marking is that it makes it easy to focus on some specific phenomenon, abstracting away from irrelevant interactions. This allows a linguist to write an experimental grammar fragment rather fast: writing the LFG grammar that models $Gen_{G_{code}}$, happens simultaneously to thinking about the constraints. So, in particular one can focus attention on a small set of relevant constraints, only generating the candidate distinctions at stake. Different hypotheses can be checked very quickly. Parts of the grammar that are not at the center of attention can be realized with a classical LFG analysis.

Of course, the fact that the grammar writer herself/himself can decide which constraints to check in which rule bears a certain risk, especially when the fragment grows over time: important interactions may be misjudged as irrelevant and thus left out. Later decisions are then set against an incorrectly biased background, and it may get hard to keep the entire constraint system under control. This is a familiar risk in grammar writing, occurring whenever some underlying generalizations are not made explicit in the grammar code. So the selective strategy of constraint checking is presumably best applied to small or medium-sized grammar fragments of theoretical interest. For larger fragments, the learnability of OT systems based on empirical data should be exploited and thus the space of candidate variance should not be restricted too much by manual preselection.

4.5 Faithfulness constraints

In sec. 4.4, OT constraints were discussed, with a limitation to those constraints that can be checked on the candidate analysis alone (i.e., without reference to the input). This seems to exclude faithfulness constraints, which are violated exactly in those cases where the candidate analysis diverges from the input. Furthermore, the restriction of the candidate generation function $Gen_{G_{code}}$ made in sec. 4.3 (definition (94) is repeated below for convenience) seems to preclude (overly) the two c-structural positions simultaneously (like split NPs in German, according to the analysis of Kuhn (1999a, 2001d)), the description-by-analysis formulation is satisfied, whereas the codescription formulation is violated. It depends on the linguistic account which variant meets the intentions.

---

85 With this strategy, it was relatively easy to implement the relevant aspects of the OT fragment of (Bresnan, 2000, sec. 2) in the XLE system, i.e., leaving aside instances of constraint violation where they were obviously irrelevant.

86 See Kuhn and Rohrer (1997), Kuhn (1998, 1999b), Butt et al. (1999a,b), King et al. (2000) for relevant discussion.
The formalization of OT Syntax in the LFG framework

unfaithful candidates from getting into the candidate set in the first place: by definition, all candidates are f-structurally subsumed by the input f-structure and may not add any semantically relevant information.

(94) Restricted definition of Gen
\[ \text{Gen}_{\text{Gen}, \text{f-str}}(\Phi_{in}) = \{ (T, \Phi') \in L(G_{unsol}) | \Phi_{in} \subseteq \Phi', \text{where} \]
\[ \Phi' \text{ contains no more semantic information than } \Phi_{in} \} \]

Excluding unfaithful candidate in Gen_{Gen,f-str}—even if it was just for the massively unfaithful candidates—would go against the methodological principle (25), according to which as much as possible should be explained as a consequence of constraint interaction. The fact that overly unfaithful candidates play no role when it comes to finding the most harmonic candidate should follow from constraint interaction alone. So, Gen_{Gen,f-str} should provide arbitrarily serious faithfulness violations. As discussed in sec. 3.2.4, the candidate set for Ann laughed should for example contain the following candidate strings (and infinitely many more):

(43) a. Ann laughed
   b. Ann did laugh
   c. it laughed Ann
   d. laughed
   e. Ann
   f.
   g. she laughed
   h. she did
   i. Ann yawned
   j. John yawned
   k. Ann saw him, etc.

As I will show in this section, the restriction of Gen_{Gen,f-str} is indeed compatible with the intuitive concept of unfaithfulness in syntax, as discussed in sec. 3.2.3. Moreover, this restriction makes redundant a dependence of faithfulness constraints on the input (besides the candidate analysis), so the form of constraints introduced in sec. 4.4 encompasses both markedness constraints and what is intuitively regarded as faithfulness constraints.

The last point involves a terminological issue, which I would like to clear up in advance: it is conceivable to define faithfulness constraints
4.5 Faithfulness constraints

as opposed to markedness constraints by their reference to the input. Under this terminology the point of this section is to show that the representations assumed in syntax work best with a system employing no faithfulness constraints at all. However, I will keep up the intuitive terminology where a faithfulness violation occurs when the surface form diverges from the underlying form, not implying that the constraints do really access the input representation.

4.5.1 Faithfulness and the subsumption-based conception of \( \text{Gen}_{G_{\text{in}}.d} \)

Definition (94) looks very restrictive, with the subsumption condition disallowing the deletion of input information (as seems to be required for modelling MAX-IO violations, cf. (38), repeated here), and an additional clause excluding the addition of semantic information (cf. DEP-IO violations/epenthesis, cf. (31)).

(38) Dropped pronominal in Italian
   a. He has sung
   b. _ ha cantato
      has sung

(31) Expletive do in English
   a. Who did John see
   b. Wen sah John
      whom saw John

However, the restrictive definition of \( \text{Gen}_{G_{\text{in}}.d} \) has at least two motivations: learnability (discussed in sec. 3.3) and computational complexity (which will be discussed further in chapter 6). Hence, it would be problematic to relax the restriction in order to permit the generation of unfaithful candidates. But the intended faithfulness violations can indeed be captured within the limits of this definition, by regarding unfaithfulness as a tension between f-structure and the categorial/lexical realization: At f-structure, semantic information may neither be added nor removed (thus the interpretation of all candidates is identical, which is important for the learner to rely on). C-structure on the other hand may contain material without an f-structure reflex (epenthesis), or leave f-structure information categorically unrealized (deletion). In the setup of LFG, this possibility can be conveniently located in the (morpho-)lexical annotations.
The formalization of OT Syntax in the LFG framework

The lexical f-annotations specify semantic and morphosyntactic information. (132) shows the lexicon entry for (the full verb) *did* with two equations in the f-annotations.

(132) \( \text{did} \ V \ * \ (\uparrow\text{PRED}) = \text{'do'} \)
\( (\uparrow\text{TNS}) = \text{PAST} \)

In analysis trees, the lexical f-annotations are sometimes shown below the phonological/orthographic forms for the terminal symbols (cf. (71)). Since they convey the morphological and lexical information, these f-annotations are called the 'morpholexical constraints' in Bresnan’s 2000 terminology (note that the term constraints is not used in the OT sense of violable constraints here). Standardly, these functional annotations are treated exactly the same way as annotations in grammar rules. This means that after instantiation of the metavariables (\( \uparrow \)) they include, they contribute to the overall set of f-descriptions the minimal model of which is the f-structure.

As Bresnan (2000) discusses for the expletive *do*, the DEP-IO-violating use of a lexical item can be modelled by assuming that (part of) its morpholexical contribution is not actually used in the construction of the f-structure. In the examples to follow, I illustrate this by encircling the respective morpholexical constraint. The ways of checking such a violation technically are discussed in sec. 4.5.2.

(133) **Violation of DEP-IO**

\[ \begin{array}{c}
\text{NP} \rightarrow \text{FP} \\
\text{NP} \rightarrow \text{IT} \\
\text{VP} \rightarrow \text{FP} \\
\text{COMP} \rightarrow \text{PRED} \\
\text{PRED} \rightarrow \text{'seem'(u)} \\
\text{PRED} \rightarrow \text{'sing'(x)} \\
\text{PRED} \rightarrow \text{'Maria'[,x]}_u \\
\end{array} \]

(133) is an example of an expletive use of the pronoun *it* in English, as assumed, e.g., in (Grimshaw and Samek-Lodovici, 1998, sec. 4). (134) is the well-known example of the expletive *do*.\(^{87}\) Note that in contrast to classical LFG, in both these cases the ordinary lexicon entry

\^{87}In these examples, I use the category symbol FP for the functional categories, rather than a a concrete instance like IP. This is meant to suggest that in principle, arbitrarily
4.5 Faithfulness constraints

is used, i.e., referential it, and full verb do. They are just used in an unfaithful way. (In classical LFG, special lexicon entries are assumed that do not introduce their own PRED value.) In Grimshaw’s terminology this type of faithfulness violation is also referred to as a case of unparsing a lexical item’s lexical conceptual structure.

MAX-IO violations are the opposite situation. Some part of the f-structure (reflecting the input) is not being contributed by any of the lexical items’ morpholexical constraints. In the examples, this is highlighted by circling the respective part of the f-structure.

(135) Violation of MAX-IO (29)

(135) is a pro-drop example from Italian. Note that—again as opposed...
The formalization of OT Syntax in the LFG framework

to classical LFG—the \textsc{pred} value of the subject is not introduced by the inflection on the verb; it simply arises “from nothing” as a faithfulness violation.

With such MAX-IO violations being part of the candidate space, it becomes conceivable to set up an OT account of ellipsis that explains the (im)possibility of ellipsis in context as an effect of constraint interaction. Let us look at the candidate (136) as one such MAX-IO-faithful candidate. It is the c-structure/f-structure analysis assumed for B’s reply in dialogue (137).\textsuperscript{88}

(136) MAX-IO-unfaithful candidate in an ellipsis account

(137) A: John claimed that Bill saw Sue.
B: And Ann.

This example is interesting since it illustrates the need for arbitrarily large portions of dropped material (the recursive embedding in A’s utterance could be arbitrarily deep, which would have to be reflected in the f-structure for B’s reply, according to the account assumed here).

Note the non-branching dominance chain dominating Ann in (136). With the re-occurrence of the categories, this structure would be ex-

\textsuperscript{88}The representation builds on L. Levin’s 1982 analysis of sluicing, assuming that at f-structure, the antecedent structure is fully reflected.
cluded by the offline parsability condition in classical LFG (77).\textsuperscript{89} So, if we want to model the ellipsis data with such representations, constraining the amount of context-recovered information as an effect of constraint interaction, the grammar $G_{\text{enviol}}$ defining the possible candidate analyses cannot be subjected to the offline parsability condition. This will fix the choice (88a) discussed in sec. 4.2.2 and 4.2.4. We get an LFG-style grammar that is not strictly an LFG grammar, producing a superset of analyses. The additional analyses do not produce any new terminal strings, but they provide strings already covered with infinitely many new f-structural interpretations (as required for the ellipsis analysis).

Giving up the classical offline parsability condition poses questions about the decidability of the processing tasks. This is discussed in chapter 6. In essence, the restricting effect of the OT constraints can be exploited to control the set of candidates that have to be effectively generated to ensure that the optimal one is among them.

Before moving on to a more rigorous formalization of faithfulness constraints, a few remarks are due on the constraint set required for the ellipsis analysis just hinted at: It is quite clear how we can make the candidate in (136) win over less elliptical competitors like and that Bill saw Ann, or even and John claimed that Bill saw Ann: the assumption of an Economy-of-expression constraint like *STRUCT outranking MAX-IO will do the job—the elliptical utterance is as expressive, using less c-structural material. However, this immediately raises the question how to make sure that Economy of expression does not fire all the time, wiping out most if not all of the linguistic material. Intuitively it is quite clear that only contextually recoverable material may be ellided, but this idea has to be implemented more formally. A rather straightforward way is to assume a constraint REC that is violated when some material is left unrealized without there being an antecedent in the local context (cf. Pesetsky (1998)). Note that the architecture of the OT system has to be extended in order to make the extra-sentential context visible for the OT constraints (a similar modification would be required in other approaches to capture recoverability too).\textsuperscript{90} The role that the context-representation plays in an OT analysis with arbitrary MAX-IO violations is discussed further in sec. 6.3.2, under a computational perspective.

\textsuperscript{89}This is so independent of a modification of offline parsability that is usually assumed LFG (compare (237) on page 211 below).

\textsuperscript{90}The condition that the REC constraint checks for is rather complicated, so one may hope to replace it by simpler, interacting constraints. This becomes possible in a bidirectional optimization framework as discussed in chapter 5, in particular in sec. 5.3.4.
The formalization of OT Syntax in the LFG framework

To sum up this subsection, the intuitive way of looking at the relation between the input and the candidates in OT-LFG should be as follows: What is characteristic of an individual candidate is its lexical material and c-structure; a candidate’s f-structure is mostly a reflex of the input.\footnote{In particular, faithfulness violations cannot lead to the situation that a candidate has a different meaning than the meaning encoded in the input. (Compare the discussion of the ineffability account of Legendre et al. 1998 in sec. 3.3.3 and 3.3.4, which works with LF-unfaithful winners. Ineffability is however derivable through bidirectional optimization, without assuming LF-unfaithfulness, cf. sec. 5.3.5.)}

Input-output faithfulness amounts to comparing a candidate’s f-structure with its morpholexical constraints. Thus one may call this the “lexiclist view of faithfulness” (cf. Kuhn (2001c)).

4.5.2 The formalization of faithfulness constraints

There are various ways how the idea discussed in the previous subsection can be formalized more rigorously. Bresnan’s 2000 original proposal is based on a special way of instantiating the metavariable \( \uparrow \) in morpholexical constraints. Classically, all metavariables \( \uparrow \) in the set of morpholexical constraints introduced by a given lexical entry have to be instantiated to the same f-structure—the one projected from the lexical item’s category. For the OT-LFG model, Bresnan assumes that some of the metavariables may be instantiated by an element that does not occur in the candidate’s f-structure—at the cost of a faithfulness violation. In order to facilitate the formulation of constraints according to the description-by-analysis account discussed in sec. 4.4.5, I proposed in (Kuhn, 2001c, sec. 3.3.2) a formalization of morpholexical constraints using a special \( \lambda \)-projection from c-structure to a special f-structure, which I will go through in the following.\footnote{In a codescription account, the formulation for the \texttt{DEP-IO} constraint would be rather straightforward: all morpholexical constraints can be assumed to be wrapped in a two-way disjunction – either make use of the morpholexical constraint or accept a constraint violation mark. The entry for \texttt{did} would thus look as follows:

\begin{verbatim}
(i) did V * ( ( \texttt{PRED} = 'do' \\
| *DEP-IO \texttt{\in} o } \\
| ( \texttt{TNS} = PAST \\
| *DEP-IO \texttt{\in} o }
\end{verbatim}

The \( \lambda \)-projection-based formulation: \texttt{DEP-IO}

I will assume that all morpholexical constraints for a lexical item will be introduced in a separate feature structure projected from the pre-
4.5 Faithfulness constraints

terminal node. Let us call this new projection the $\lambda$-projection (for “lexical”), describing a correspondence between c-structure and “l-structure”. When faithfulness is met, all elements in this set will subsume the f-structure projected from the same category; however, for unparsed morpholexical constraints, a mismatch occurs.

In (138), (a modification of (Bresnan, 2000, (44))), the idea is illustrated for an example violating faithfulness: the PRED constraint introduced by did does not re-appear in the f-structure.

(138) **DEP-IO violation (with PRED ‘do’ missing in f-structure)**

\[
\begin{array}{c}
\text{[...]} \xleftarrow{\lambda} \text{IP} \\
\begin{array}{c}
PRED \text{ ‘do’} \\
TNS \text{ PAST}
\end{array} \\
\begin{array}{c}
PRED \text{ ‘say’} \\
\text{SUBJ} \\
\text{OBJ} \\
TNS \text{ PAST}
\end{array} \\
\end{array}
\]

In order to reach the intended effect, the lexicon entries have to look as follows (recall that $\uparrow$ is short for $\phi(M\ast)$ – i.e., the f-structure projected from the current node’s mother, in this case the pre-terminal):

(139) **did I * (f$_1$ PRED) = ‘do’**

\[
\begin{align*}
&f_1 = \lambda(M\ast) \\
&(f_1 = \uparrow) \\
&(f_2 \text{ TNS}) = \text{PAST} \\
&f_2 = \lambda(M\ast) \\
&(f_2 = \uparrow)
\end{align*}
\]

93In (Kuhn, 2001c, sec. 3.3.2), a set of feature structures was assumed in the l-structure, containing a small feature structure for each morpholexical constraint. The faithfulness constraints were then checked for each of the small feature structures. Here, I assume that the faithfulness constraints are checked for the atomic values, so it is unnecessary to assume this additional complication in the geometry of l-structure.
The formalization of OT Syntax in the LFG framework

\[\text{say } V \ast (f_1 \text{pred}) = 'say'\]
\[f_1 = \lambda (\mathcal{M} \ast)\]
\[(f_1 = \uparrow)\]

For every morpholexical constraint, there are three annotation schemata, making use of a distinct local metavariable referring to a feature structure \((f_1, f_2, \ldots)\). The three schemata are: (i) the lexical constraint itself,\(^94\) (ii) an \(f\)-equation introducing the morpholexical constraint to the \(l\)-structure projected from the pre-terminal, and (iii) an optional \(f\)-equation introducing the constraint at the level of \(f\)-structure. The optionality of schema (iii) leads to the presence of unfaithful analyses.

The faithfulness constraint \(\text{DEP-IO}\) can now be formulated as follows:\(^95\)

\[(140) \quad \text{DEP-IO}\]
\[\forall n, P. \left[ \left( \text{atomic-f-str}(\ast) \land \text{cat}(n) \land (\lambda (n) P) = \ast \right) \rightarrow (\phi(n) P) = \ast \right]\]

“For all categories \(n\) and feature paths \(P\), if \(\ast\) is an atomic value under \(P\) in the \(\lambda\)-projection from \(n\), then \(\ast\) is also the value under \(P\) in the \(\phi\)-projection from \(n\).”

Since the metavariable \(\ast\) is generally instantiated to every structural element, it is now in particular instantiated to the feature structures in \(l\)-structure. Note that the value ‘do’ of the \(\text{pred}\) feature in (138) fails to satisfy this constraint: instantiating \(n\) as the I category and feature path \(P\) as \(\text{pred}\), we have \((\lambda(I) \text{pred}) = 'do'\), but not \((\phi(I) \text{pred}) = 'do'\).

An issue not addressed so far is the following: What controls the choice of expletive elements \((\text{do} \text{ rather than } \text{shout etc.)}\)? This question is briefly addressed by (Bresnan, 2000, sec. 2), adopting the basic idea from (Grimshaw, 1997, 386): the assumption is that for a verb like \(\text{do}\), “[t]he unparsing of its semantically minimal \(\text{pred}\) feature is a smaller

\(^{94}\)Note that the assumption is that all constraints are expressed as defining equations, rather than constraining equations, and that Completeness and Coherence are checked only on \(f\)-structure, not on \(l\)-structure.

\(^{95}\)This formulation contains a universal quantification over category nodes and features paths, as was excluded in sec. 4.4.4. However, the domain of the variables is restricted by the (finite) lexicon, so a reformulation avoiding universal quantification would be possible (using the metavariable \(\ast\) for the category which is being quantified over in (140)). A parametrization of the constraint to a particular type of information as is usually assumed—\(\text{DEP-IO(feature)}\)—is straightforward too:

\[\forall n. \left[ \text{atomic-f-str}(\ast) \land \text{cat}(n) \land (\lambda(n) \text{feature} = \ast) \rightarrow (\phi(n) \text{feature}) = \ast \right]\]
4.5 Faithfulness constraints

violation of faithfulness than that incurred by unparsing the semantically richer PREDs of *shout*, *obfuscate*, or any other verb in the English lexicon."

For concreteness, let us assume that this intuition is modelled by a conceptual hierarchy of PRED values—or lexical conceptual structures. More specific sub-concepts will inherit all the information from their super-concepts, plus they will add some information. Now, to evaluate faithfulness constraints on “unparsed” PRED values, the conceptual contribution they would have made is considered piece by piece, i.e., concepts embedded more deeply in the concept hierarchy will incur more violations than the more general ones. In effect, everything else being equal, the most general available concept will be picked as an expletive element.96 I will not pursue this issue further in this book.

The Max-IO constraint

For the Max-IO we get a symmetrical picture as with Dep-IO. An example is given in the structure (141) for Italian *ha cantato* (has sung). Note that none of the λ-projected (i.e., morpholexical) feature structures introduces the PRED values under SUBJ, which does appear in the f-structure.

96To turn this idea into an account with reasonable empirical coverage, it clearly has to be complemented by some additional device allowing for conventionalization of the use of a particular lexical item out of a choice of semantically very similar items. For example, most Romance languages use the verb derived from Latin *habere* (‘have’) as a perfect auxiliary, while Portuguese uses the verb derived from tenere (‘hold’).
The formalization of OT Syntax in the LFG framework

For technically “assembling” the f-structure, the grammar \( G_{envol} \) has to provide a way of optionally introducing “pseudo-lexical constraints” for each piece of information occurring in the input. There are several ways in which this could be done. One is to have a pseudo-lexical annotation in the grammar rule for the root symbol, using functional uncertainty to reach arbitrary embedded f-structures and optionally provide some missing information, such as the \( \text{pred} \)-value ‘pro’. If we have a recursive rule for the root symbol, this way of adding information can be used over and over again:

\[
\text{(142) Gen grammar rule, with the potential of providing pseudo-lexical constraints}
\]

\[
\text{ROOT} \rightarrow \left\{ \begin{array}{l}
\text{ROOT} \\
(\uparrow_{\text{GF}^* \text{pred}}) = \text{‘pro’} \\
\vdots
\end{array} \right\}
\]

Alternatively, pseudo-lexical constraint introduction without the functional uncertainty could be foreseen for all maximal projections, by adding a recursion (this would change the c-structure representation, but the original structure can be systematically recovered if an appropriate marking is used):

\[
\text{(143) Pseudo-lexical constraint introduction at the level of maximal categories}
\]

\[
\text{XP} \rightarrow \left\{ \begin{array}{l}
\text{XP} \\
(\uparrow_{\text{pred}}) = \ldots \\
\vdots
\end{array} \right\}
\]

Such pseudo-lexical constraints can of course be used only at the cost of incurring a \( \text{MAX-IO} \) violation. In example (141), we made use of this option for the \( \text{pred} \)-value under \( \text{SUBJ} \). We can formalize \( \text{MAX-IO} \) as follows:\textsuperscript{97}

\[
\text{(144) MAX-IO}
\]

\[
\text{atomic-f-str}(\ast) \rightarrow \exists n, P. [\text{cat}(n) \land (\phi(n) P) = \ast \land (\lambda(n) P) = \ast]
\]

“If \( \ast \) is an atomic value then there is some category \( n \), such that \( \ast \) is embedded under some path \( P \) in the \( \phi \)-projection from \( n \) and \( \ast \) is also the value under \( P \) in the \( \lambda \)-projection from \( n \)”.

Again one can check that this constraint is violated in (141): for the value ‘pro’ embedded in the f-structure under the path \( \text{SUBJ \ pred} \), we

\textsuperscript{97}Again a parametrization \( \text{MAX-IO} (\text{FEATURE}_i) \) is possible (compare footnote 95):

\[
\text{atomic-f-str}(\ast) \rightarrow \exists n. [\text{cat}(n) \land (\phi(n) \ \text{FEATURE}_i) = \ast \land (\lambda(n) \ \text{FEATURE}_i) = \ast]
\]

Note that universal quantification over c-structure nodes would not give us the right result since \( \phi \) may map several nodes to the same f-structure, and it is sufficient if \( \ast \) was introduced lexically by one of them.

120
4.6 Summary

do not find any category such that it is embedded in this category’s l-structure under the same path as in f-structure (in particular, we do not find the value ‘pro’ under \textit{SUBJ PRED} in \lambda(I)).

Conclusion

Concluding the section on faithfulness constraints, we can note that the subsumption-based conception of \textit{Gen}_{\infinitive} is indeed compatible with the idea of arbitrarily heavy faithfulness violations: the unfaithfulness arises as a tension within the LFG candidate analyses, since the categorial/lexical structuring need not reflect the f-structure faithfully. As a consequence of this conception, even the faithfulness constraints can be checked on the candidate analyses alone, without explicit reference to the input.\footnote{Compare (Heck et al., 2000) for a similar result, and the discussion in footnote 36 on page 53.} Both types of faithfulness violations raise certain issues for the processing which will be discussed in chapter 6.

4.6 Summary

In this chapter, I proposed a formalization of OT syntax in the formal framework of LFG, elaborating the original ideas of Bresnan (1996, 2000). This formalization meets the requirements developed in chapter 3 under empirical and conceptual considerations: the cross-linguistic variance in surface realization of underlying arguments can be derived as an effect of constraint interaction. At the same time, the precondition for learnability is met, since the semantically (and pragmatically) interpreted part of the candidate representations is identical for all members of a candidate set.

This was reached by assuming non-derivational candidate analyses based on LFG’s system of correspondence between parallel structures (most notably c- and f-structure). Defining possible candidate analyses as the structures produced by a formal LFG-style grammar (G_{\text{inviol}}) comprising the inviolable principles, the entire OT system can be defined in a declarative, non-derivational fashion. The only purpose of the input (or Index) is the definition of candidate sets; since the input fixes the interpretation of the candidates, the formal representation used for the input is that of a partially specified f-structure.

The formal relation between the input f-structure and the candidates in the candidate set defined by this input is subsumption. The input
subsumes the candidate’s f-structure, and at the same time the candidate may not specify additional semantic information. Thus, the candidates differ only in terms of c-structural information and f-structural information insofar as it is not semantically relevant (i.e., purely morphosyntactic feature information). Despite this limitation as to the degree of formal divergence between input and candidates, all empirically motivated cases of faithfulness violation can be modelled. Faithfulness emerges as a candidate-internal concept and can be checked by comparing a candidate’s f-structure and the morpholexical specification of the lexical items used.

Thus, it is sufficient for both markedness constraints and faithfulness constraints to refer to the candidate structure exclusively. Constraints are formulated as schemata in a tree/feature description logic over LFG representations. For constraint checking/marking, the schemata are instantiated to all structural elements (c-/f-structures) in a candidate analysis. Every structural element for which a constraint is not satisfied increases the constraint violation count for this constraint. The final evaluation step based on the constraint counts for all analyses in the candidate set is the canonical harmony-based evaluation step of OT with no specific modification for the OT-LFG scenario.

The diagram in (145) illustrates the formal setup graphically. The broken lines suggest that the relations between the formal elements making up the OT-LFG systems should be seen in a declarative way, as opposed to derivational processes.

Ultimately, it is important to note that the language generated by an OT-LFG system is defined with an existential quantification over underlying inputs. The definition is repeated here:

\begin{equation}
\text{(87) Definition of the language generated by an OT-LFG system}
\end{equation}

\[
O = \langle G_{\text{inviol}}, (\mathcal{C}, \gg_{\mathcal{L}}) \rangle
\]

\[
L(O) = \{ \langle T_j, \Phi_j \rangle \in L(G_{\text{inviol}}) \mid \exists \Phi_{in} : \langle T_j, \Phi_j \rangle \in \text{Eval}_{(\mathcal{C}, \gg_{\mathcal{L}})}(\text{Gen}_{G_{\text{inviol}}}(\Phi_{in})) \}\}
\]

So, given an LFG-style grammar $G_{\text{inviol}}$ for the inviolable principles and a set of constraints $\mathcal{C}$ with a language-specific ranking $\gg_{\mathcal{L}}$, the set of grammatical analyses is defined as those analyses $\langle T_j, \Phi_j \rangle$ produced by $G_{\text{inviol}}$, for which there exists an underlying input $\Phi_{in}$ such that $\langle T_j, \Phi_j \rangle$ is optimal (based on $\mathcal{C}$ and $\gg_{\mathcal{L}}$) in the candidate set defined by $\Phi_{in}$.
4.6 Summary

The OT-LFG setup

Input/Index:
Partial f-structure

Subsumption

Cand_1

C'

\langle n_1^1, n_1^2, n_1^3 \ldots n_1^k \rangle

Cand_2

C'

\langle n_2^1, n_2^2, n_2^3 \ldots n_2^k \rangle

Cand_3

C'

\langle n_3^1, n_3^2, n_3^3 \ldots n_3^k \rangle

Eval_{(c, \triangleright_\phi c)}

Optimal