Generalized Tree Descriptions for LFG

Jonas Kuhn
University of Texas at Austin
Department of Linguistics

jonask@mail.utexas.edu

Proceedings of the LFG03 Conference

Saratoga Springs, NY, USA

Miriam Butt and Tracy Holloway King (Editors)

2003
CSLI Publications
http://csli-publications.stanford.edu/

Abstract

This paper proposes a novel description format for c-structure. The explicit phrase structure rewrite rules are replaced by more general constraints in a tree description language: (weak) monadic second order logic (MSOL). Exploiting the fact that the MSOL-definable tree languages correspond to the parse trees of context-free string languages, it can be shown that the generative capacity of the LFG formalism is not altered. Applying the MSOL-based formulation of c-structure descriptions to Optimality-Theoretic LFG results in a significant clarification of the prerequisites for a computationally well-behaved Optimality-Theoretic LFG system.

1 Introduction

This paper has two main parts: in the first part (section 2), a logic-based specification scheme for LFG is proposed and discussed. The second part (section 3) sketches an application of this specification format to formal considerations within Optimality-Theoretic LFG. (Kuhn (in preparation) discusses this application in detail.)

1.1 Motivation

The point of departure for the generalized tree description format proposed here is the following: In the LFG formalism, the formally exact specification of c-structure is done by context-free rewrite rules (with generalized right-hand sides, including regular predicates like Kleene closure, etc.). Using context-free rule specification has the obvious advantage that computational results for this class can be transferred straightforwardly to the c-structure part of LFG parsing. (The regular expressions on the right-hand sides of rules can be reduced to standard context-free rules too.)

However, the rewrite-rule-based specification of c-structural regularities places fairly strong limitations on the generality of principles that are expressible from the point of view of syntactic theory. As a consequence, many theoretical generalizations over possible c-structures (and the c-structure/f-structure relation) have had no *explicit* correspondent in the formal specification of an LFG grammar. For example, the c-structure part of the IP and I′ rules is ultimately specified like in (1) or in a similar way; these rules *obey* (extended) X-bar categorial principles, but the principles are not formally expressed. In other words, the absence of nonsensical rule like (2) appears to be purely accidental.

(1) $\text{IP} \rightarrow (\{\ \text{DP} \mid \text{NP} \mid \text{PP} \mid \text{CP} \mid \text{IP} \mid \text{S}\ \})\ (\text{I}')$
$\text{I}' \rightarrow (\text{I})\ (\text{VP})$

(2) $\text{D}' \rightarrow (\{\ \text{N}' \mid \text{VP}\ \})\ \text{P}$

In (3), a number of non-trivial sets of principles are listed that have been assumed in LFG.

(3) a. X-bar categorial generalizations
(e.g.: only maximal projections are allowed in specifier of XP)
b. Endocentricity (every lexical category has an extended head)
c. Annotation principles (e.g.: specifiers of functional projections are discourse functions)
d. OT constraints like Alignment constraints

Since the actual formal c-structure specification in the LFG formalism rests on simple rewrite rules, there has been a tacit assumption: Theoretical principles generalizing over c-structure, like the ones in (3), are somehow “compiled into” the actual c-structure rules. In the long run, this situation is not fully satisfactory for a formally rigorous constraint-based/description-based approach.

There are several conceivable ways for trying to make c-structure-level generalizations formally explicit:

1. Macro/template devices in the description language
2. A meta specification scheme using (multiple) inheritance hierarchies
3. C-structure principles as filters on trees
4. A "constructive" tree description logic

The first option – the use of macro/template devices in description language – has been applied in grammar writing efforts like the ParGram project (compare (Butt et al. 1999b, sec. 13.2), Butt et al. (1999a, 2003)). The XLE grammar development system for the LFG formalism[1] provides a number of different such devices (meta categories, rule macros, f-annotation templates, parametrized rules). (4) indicates how such devices can be used to make generalizations explicit and re-usable: `MaxProj` is a meta category, `FCatSpec` (for "functional category's specifier position") is a macro that can be used in various rules.

```
(4)  MaxProj  = { CP | IP | S | DP | NP | AP }.
     FCatSpec = MaxProj: (ˆ {SUBJ|TOPIC|FOCUS})=!.
     CP  --> ( @FCatSpec )  Cbar.
     IP  --> ( @FCatSpec )  Ibar.
```

The primary motivation for these techniques comes from large-scale grammar development, where the generality of rule formulation is just one criterion which has to be traded off against robustness, coverage of rare and little studied constructions and other "grammar engineering" factors. Hence the macro devices are not necessarily the best choice for formalizing linguistic principles on theoretical grounds: While (Kuhn 1999b) shows that the concept of parametrized rules (using complex category symbols) can be exploited to implement a more general rule set for X-bar theory (compare (5) and figure 1), there are still limitations to the approach.

```
(5)  XP[_C]    -->  YP: (ˆDF)=!;   Xbar[_C]: ˆ=!.
     Xbar[_C]  -->  X[_C]: ˆ=!;    YP: (ˆGF)=!;*.
```

In particular, the situations in which the principles hold still need to be stipulated; no axiomatic formulation of the principles is provided.

The second option for expressing c-structure generalizations is the use of (multiple) inheritance hierarchies. The best-known application of this technique are the "Immediate Dominance Schemata" of Head-driven Phrase Structure Grammar (HPSG, compare (Pollard and Sag 1994, sec. 1.5)). An approach that uses some related ideas for LFG was sketched in Kuhn 1999a; Clement and Kinyon (2003) propose a meta specification scheme for LFG, based on Tree-Adjoining-Grammar (TAG). Generally, a meta specification approach requires an augmentation of the representational system in order to be able to refer to rule elements (like, for instance, the syntactic head) throughout the various principles. This is to a certain extent against the spirit of LFG, which avoids extending representations if there is a way of reaching the same explanatory effect based on more powerful *descriptions* of the same simple representations. Thus, let us explore a different alternative in the context of this paper.

As a third option, one could formalize c-structure principles as filters on c-structure trees (similar as Completeness and Coherence for f-structure). This would require a tree description language (but

[1]`www.parc.com/istl/groups/nltt/xle/`

```
X[_F,_C,_B]  -->
     { e: _F = +f                         " FP --> (GP) (F′) "
          _B = 2;
       (X[+f,any,2]: (ˆDF)=!;)          "specifier"
       (X[_F,_C,1]:  ˆ=!;)              "head"

     | e: _F = -f                         " L′ --> (L) (FP) "
          _B = 1;
       (X[_F,_C,0]:  ˆ=!;)              "head"
       (X[+f,any,2]: (ˆGF)=!;)          "complement"

     | e: _B = 1;                         " X′ --> (X) (LP) "

       (X[_F,_C,0]:  ˆ=!;)              "head"
       (X[-f,_C,2]:  ˆ=!;)              "cohead"   }.
```

Figure 1: *Implementation of general X-bar scheme using parametrized rules (Kuhn 1999b)*

note that the $\mathcal{M}$ functor for referring to a c-structure node's mother already exists in standard LFG). To my knowledge, such a c-structure fi ltering mechanism has not been explored for standard LFG. Let us here follow the forth option (which is somewhat related to the previous one): using a “constructive” logic-based specifi cation of trees instead of the explicit rewrite rules.[2]

2 Logic-based c-structure specification

2.1 The tree description logic

As an alternative to the procedurally grounded notion of context-free rewrite rules for c-structure description, let us explore the use of formulae in (Weak) Monadic Second Order Logic (MSOL) for describing the set of possible c-structure trees. MSOL is an extension of classical fi rst-order logic including variables ranging over one-place (=monadic) predicates (or, equivalently, over sets[3]) and quantifi ers over these variables. (6) illustrates the general shape of MSOL formulae (it is not particularly meaningful):

(6) $(\forall X)(\forall x, y)[x, y \in X \rightarrow (x \triangleleft^* y \vee y \triangleleft^* x)]$

Using the dominance relation $\triangleleft^*$ (which can be defi ned within MSOL, compare Rogers 1998),[4] (6) says that for all sets X of tree nodes, for any two nodes x, y from that set, either x dominates y or y dominates x (note that the $\triangleleft^*$ relation includes the reflexive case of a node dominating itself). If we used this sample formula as a general grammatical principle, it would force all trees to be somewhat degenerate since it only allows for nonbranching chains of nodes.

Let is look at an LFG example of constraints on c-structure: *Endocentricity* requires every category to have an extended head, where the concept of an extended head requires a very careful defi nition (for more background on Endocentricity compare (Bresnan 2001, sec. 7.2)). As (7) shows, the extended head requirement can be formulated straightforwardly in MSOL, using an *ExtHd* predicate, which requires further elaboration.

[2]In the context of his categorial grammar-based discussion of LFG, Muskens (2001) proposes a logic-based c-description formalism too. In the present paper however, I try to provide a formalization that differs only minimally from standard LFG.

[3]In *weak* monadic second order logic, the variables range over *finite* sets only.

[4]Furthermore, we can assume an immediate dominance relation $\triangleleft$, non-reflexive domination $\triangleleft^+$, and a precedence relation $\prec$.

(7) Endocentricity: every lexical category has an extended head

$(\forall x)[\mathit{LexCat}(x) \rightarrow [(\exists y)[\mathit{ExtHd}(y, x)]]$

In (8) we see the definition of *extended head* that (Sells 2001, 115) provides. This concept can be formalized as in (9), still leaving open how we formalize co-projection of two nodes (*CoProj*(x, y)).

(8) Extended Head (formulation of Sells 2001, 115)
X is an extended head of Y if X is a lexically filled category, X corresponds to the same f-structure as Y, and every node that dominates X and is not dominated by Y also dominates Y.

(9) MSOL definition of the extended head relation

$$\begin{aligned}\mathit{ExtHd}(x, y) \quad \equiv \quad & \mathit{LexFilled}(x) \wedge \mathit{CoProj}(x, y) \wedge \\ & (\forall z)[(z \lhd^{+} x \wedge \neg(y \lhd^{+} z)) \rightarrow z \lhd^{+} y]\end{aligned}$$

(10) $\mathit{LexFilled}(x) \equiv (\exists y)[\mathit{Terminal}(y) \wedge x \lhd^{+} y]$

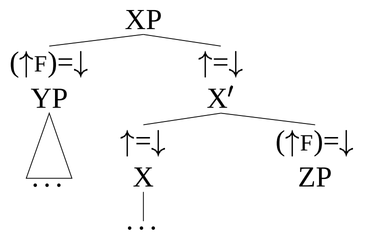
There are at least two conceivable ways of defining the *CoProj* predicate:[5]

(11) (a) $\mathit{CoProj}(x, y) \equiv \phi(x) = \phi(y)$

(b) $$\begin{aligned}\mathit{CoProj}'(x, y) \quad \equiv \quad & (\exists P)[\mathit{Connected}(P) \wedge x, y \in P \wedge \\ & (\forall z_1, z_2)[(z_1, z_2 \in P \wedge z_1 \lhd z_2) \rightarrow \phi(z_1) = \phi(z_2)]]\end{aligned}$$

Variant *CoProj*$'$ (11b) includes the additional constraint that the co-projecting c-structure nodes are connected by a sequence of c-structure nodes *all* of which are mapped to the same f-structure. This excludes the case in which mapping to the same f-structure arises as an effect of independent f-structure unification, as it may arise in the following schematic configuration:

(12)



Using the simpler *CoProj* definition (11a) in the definition of *extended head* (9) would make YP an extended head of ZP, i.e., it would make this c-structural concept even less local than one ususally assumes it to be. So, presumably (11b) is the formalization that we want.

In section 2.3, we will come back to the differences between (11a) and (11b) when we address the specification of the c-structure/f-structure relation in an MSOL-based framework. We will see that the concept as defined in (11a) exceeds the generative power of standard LFG (i.e., it could not be expressed in full generality within the standard LFG formalism); we will also see how the MSOL formulation is ruled out by straightforward restrictions on simultaneous descriptions of c-structure and f-structure.

The *extended head* example shows that a precise formulation of principles on c-structure is very important. From the prose definition in (8) it is not obvious that the literal interpretation of this definition is not expressible in standard LFG.

[5]*Connected* holds of a set of tree nodes iff it is a connected subgraph of a tree (compare e.g., (Morawietz and Cornell 1999)): $\mathit{Connected}(P) \equiv (\forall x, y, z)[(x \in P \wedge y \in P \wedge x \lhd^{+} z \wedge z \lhd^{+} y) \rightarrow z \in P]$

2.2 Some important properties of Monadic Second-order Logic

MSOL has been explored in detail in work on formal language theory (compare e.g. Gécseg and Steinby 1997). It has been used in formal work on syntactic formalisms, in particular for the formalization of heavily tree-based theories of syntax (e.g., Rogers 1997, 1998, 2001, Morawietz and Cornell 1999).

The following properties make it particularly attractive for linguistic formalisms:

- **Decidability.** This guarantees that the trees described by a formula can be effectively constructed.
- The class of tree sets definable with MSOL coincides with the class of parse trees for generated by **context-free grammars**.
- **Possibility of compiling tree automata.** The relation between MSOL expressions and trees (viewed as tree automata) is comparable to the relation between regular expressions and finite-state (string) automata. This means that logical tree descriptions can stay at a general abstract level, while it is ensured that a precise computational model can be compiled out.

The following subsections provide an illustration of the last point. To appreciate the usefulness of automata compilation, it is instructive to draw a parallel between the MSOL/tree automata relation and the relation between regular expressions and finite-state string automata, which is familiar from basic automata theory.

Excursion: regular expressions and finite-state (string) automata. Regular expressions and finite-state string automata are descriptively equivalent. So, generalizations about different dimensions of strings can be stated as separate regular (sub-)expressions, which can then be combined by regular expression operations such as intersection. Since the regular expression formalism is closed under these operations, we can be sure that a single finite-state automaton for computational application can be compiled from the resulting complex expressions.

This description technique makes it possible to formulate linguistic principles and their mutual relationship in a very general way, not having to worry about details of interaction in the formulation of the principles. The "compilation" into a usable computational device can be taken care of in a general mechanism.

A simple abstract example for a regular expression involving the intersection of two "principles" is given in (13).[6] (14) shows the compiled-out automaton (which in this case is extremely simple, so the reader can verify that intersection of the regular expressions has the effect of producing the same language).

(13) *Regular expression:*
$[(a^*|a\ b)\ c]\ \&\ [?^*\ b\ ?^*]$

(14) *Finite-state automaton:*

$$\begin{tikzcd} {} \arrow[r] & q_0 \arrow[r, bend left, "a"] & q_1 \arrow[r, bend left, "b"] & q_2 \arrow[r, bend left, "c"] & \boxed{q_3} \end{tikzcd}$$

[6]The '?' in the regular expression refers to an arbitrary symbol; '&' is the intersection operation, '|' is union/disjunction. '$*$' is the Kleene star for zero to n repetitions, the brackets are used for grouping.

MSOL and finite-state tree automata. Having seen the regular expression/finite-state string automata relation, let us look at MSOL formulae and their relation to a generalization of finite-state automata: finite-state tree automata. Like in the previous situation, we find that (complex) formulae in MSOL are descriptively equivalent to tree automata. (We cannot go into a proof of this relation – the reader is referred to Gécseg and Steinby 1997, for example.)

Let us look at tree automata a little more closely. There are various sub-types of finite-state tree automata. For our purposes it suffices to consider *bottom-up tree automata*, which are defined as in (15).

(15) *Definition: Bottom-up finite-state tree automaton*

$\langle A, \Sigma, a_0, F, \alpha \rangle$:

A:	finite set of states;
Σ:	alphabet of node labels (including terminals and nonterminals);
$a_0 \in A$:	initial state;
$F \subseteq A$:	set of final states;
α:	state transition function $\alpha(b_1, \dots, b_k, N) = b'$, where $k \geq 0, b_1, \dots, b_k, b' \in A, N \in \Sigma$; $b_1, \dots, b_k$ are the present states for k daughters of a node with label N; b' is the new state of the automaton.

In figure 2, an example bottom-up tree automaton is seen, including a sample derivation for the tree we get for *Ann knew Bill left*.

$A = \{a_0, a_1, a_2, a_3, a_4, a_5, a_6\}$;
$\Sigma = \{$S, NP, VP, V, Ann, Bill, knew, left$\}$;
$F = \{a_6\}$;
$\alpha(a_0,\text{Ann}) = a_1$,
$\alpha(a_0,\text{Bill}) = a_1$,
$\alpha(a_0,\text{knew}) = a_2$,
$\alpha(a_0,\text{left}) = a_2$,
$\alpha(a_1,\text{NP}) = a_3$,
$\alpha(a_2,\text{V}) = a_4$,
$\alpha(a_4,\text{VP}) = a_5$,
$\alpha(a_3, a_5,\text{S}) = a_6$,
$\alpha(a_4, a_6,\text{VP}) = a_5$.

Sample derivation

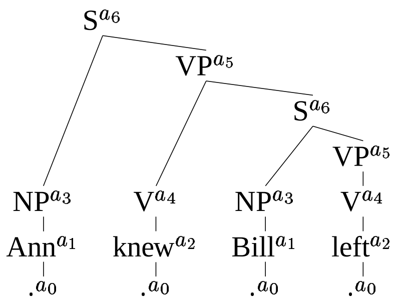


Figure 2: *Example of a bottom-up finite-state tree automaton, including a sample derivation*

A bottom-up tree automaton is very much like classical finite-state automata accepting the possible strings of terminal/nonterminal symbols *on a tree branch from leaf to root*. (So we could envisage a classical automaton traversing the left-most branch in the tree in figure 2, accepting 'Ann NP S', as it goes through states a_0, a_1, a_3, a_6. The classical finite-state automaton would have transitions $\alpha(a_0,\text{Ann}) = a_1$, $\alpha(a_1,\text{NP}) = a_3$, and $\alpha(a_3,\text{S}) = a_6$ (!).) What is special about tree automata is that we have to think of them as initially running several such branch acceptance processes in parallel. As we move from the leaves towards the root, more and more of the parallel processes get merged, using a special type of state transition $\alpha(b_1, \dots, b_k, N) = b'$: in such a transition, k "subprocesses" (which have to be in state b_1 through b_k respectively) are merged into a single process, reading (or writing) a nonterminal symbol N. The resulting single process is then in state b'. The effect of this process merger is to accept a local subtree with mother category N and k daughters each of which has already been accepted bottom-up. So the sample tree diagram in figure 2 should be read as a trace of four parallel subprocesses, each starting in state a_0 at one of the leaves, which are then merged at various levels. The single remaining process ends in final state a_6, which is a sign that the tree we see is accepted by the tree automaton.

Due to the descriptive equivalence of MSOL and tree automata, we can assume a mechanical procedure that will produce tree automata for any MSOL formulae (i.e., in particular the complex conjunction of all grammatical principles that we want to assume). That means we do not actually have to think about which state transitions we will need in order to keep track of certain non-trivial dependencies between tree nodes (such as the extended head relation discussed above).

As pointed out above, this is very similar to the process of compiling complex finite-state automata from a linguistically perspicuous collection of regular expressions, as it is familiar from applications of finite-state technology for morphological analysis or shallow syntactic grammars.

2.3 The relation between c-structure description and f-structure description

If we want to apply MSOL for the description of LFG's c-structure we also need to adjust the expression of f-structure constraints, which are classically provided as annotations in the rewrite rules of an LFG grammar.

There are at least two possibilities:

1. We could use MSOL also to specify f-structure and c-structure/f-structure correspondence. MSOL can be used to describe graph languages and the relation between two graphs.

2. We could keep up standard LFG concept of f-annotations, using the standard ϕ-projection and making only reference to the node itself or its mother node (i.e., using only $\uparrow$ and $\downarrow$ relative to a given c-structure node).

The second option will lead to a hybrid system of MSOL formulae and subexpressions in the standard LFG feature logic.

Although the first option avoids a hybrid specification system and is thus presumably superior on aesthetic grounds, let us for now follow the second approach. The advantages of this second choice are that the specifics of the LFG f-description language can be directly inherited: the distinction between defining and constraining equations, the special interpretation of disjunction and negation, the treatment of functional uncertainty, closed sets, etc. All these characteristic would have to be reconsidered if we followed the first option, potentially leading to a notion of f-structure with quite a different character. For the second option, which is a minimal alteration of the original LFG formalism, it is fairly straightforward to show expressive equivalence with standard LFG, as I will indicate in section 2.4.

Having decided that f-descriptions will continue to be in the form of annotations to c-structure categories, we need a way of combining them with the new logic-based tree descriptions. In LFG-grammatical MSOL formulae, *f-annotation predicates* will have a special status: (16). They can either be one-place predicates, describing just the f-structure projected from a single node, or they can be two-place referring to a node and its c-structural mother.

(16) The notion of *f-annotation predicates in MSOL formulae*

a. Tree descriptions can only make indirect reference to f-structure: by including f-annotation predicates

b. In the definition of f-annotation predicates, the only reference to tree node variables may be as the argument of the ϕ projection (and possibly other projections).

Examples for f-annotation predicates and their definition in terms of classical LFG feature logic are given in (17).

(17)	a.	*NumSing*(x)	$\equiv$	$(\phi(x)\text{NUM}) = \text{SING}$
	b.	*SubjEmbed*(x, y)	$\equiv$	$(\phi(x)\ \text{SUBJ}) = \phi(y)$
	c.	*AdjunctEmbed*(x, y)	$\equiv$	$\phi(y) \in (\phi(x)\ \text{ADJUNCT})$
	d.	*CoHead*(x, y)	$\equiv$	$\phi(x) = \phi(y)$

For MSOL-based tree description, the f-annotation predicates are treated as unanalyzed assertions, similar to other descriptive category predicates like *Nominal*(x). The definitions in (17) are not resolved until a trees have been constructed according to the tree description predicates.

Using only $\phi(*)$ and $\phi(\mathcal{M}*)$ references in f-annotation predicates. An additional condition on the two-place f-annotation predicates has been left implicit so far. In standard LFG f-annotations, we normally refer to the node's f-structure and the mother node's f-structure only. For $\phi(*)$ and $\phi(\mathcal{M}*)$ there are the short-hand metavariables $\downarrow$ and $\uparrow$. If we want to keep up this restriction, we should assume the following meta constraint on f-annotation predicates:

(18) Meta constraint on two-place f-annotation predicates P:
$P(x, y) \Rightarrow x \triangleleft^{+} y$

As a consequence of this, more general f-annotation predicates spanning larger tree-structural configurations are excluded. In particular, this excludes the seemingly simpler version of the *CoProj* predicate (11a) discussed in section 2.1. Version (11b) on the other hand can easily be made compatible with this restriction (and restriction (16)), by replacing the final equation $(\phi(z_1) = \phi(z_2))$ with a call of *CoHead* as defined in (17): *CoHead*(z_1, z_2)

As it turns out, there is no way of encoding the effect of the (11a)-based formulation in full generality using the f-annotations of a classical LFG grammar. It is not possible to keep track of arbitrarily many non-local coprojection configurations. So, it is appropriate to rule out (11a) by the restrictions in this section.

Definition of the c-structure/f-structure analyses in MSOL-based LFG. We can now see how the valid c-structure/f-structure analyses for a given string has to be defined based on the new description format. There are two steps:

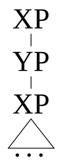
1. The set of c-structure candidates is obtained by constructing the set of trees that are a model for the MSOL grammar formula, conjoined with an MSOL description of the terminal string.[7] The f-annotation predicates are ignored in this step (i.e., they are assumed to be satisfiable).

2. For each c-structure candidate, the f-annotation predicate terms from all tree nodes are conjoined and resolved, instantiating the node variables to particular nodes from the tree model; then the f-constraints are resolved as in standard LFG. The Completeness and Coherence condition are applied as normal.

[7]We will shortly address one additional condition.

Note that of course, step 2. may exclude many (or all) of the original trees, to the extent that they do not support any consistent (and complete/coherent) f-structure.

The set of c-structure candidates constructed in step 1. may be infinite in the general case. This is analogous to the situation in standard LFG, where a context-free grammar could generate an infinite number of trees over a given string. By assuming the non-branching dominance condition (offline parsability; Kaplan and Bresnan 1982, 266), the number of actual c-structure candidates is reduced to finitely many, excluding configurations like the following:

(19) *


The non-branching dominance condition (offline parsability) can be expressed as an MSOL axiom schema:

(20) Non-branching dominance axiom schema

$$(\forall X) \quad [\mathit{NonBranchChain}(X) \\ \rightarrow (\forall x, y)[x, y \in X \wedge (\ \textstyle\bigwedge_{P \in \mathbf{P}} P(x) \leftrightarrow P(y)) \rightarrow x = y]]$$

where

- **P** is the set of all category and f-annotation labels used in the grammar formula
- $\mathit{NonBranchChain}(X) \equiv \quad (\forall x, y)[x, y \in X \rightarrow (x \lhd^* y \vee y \lhd^* x)]$
  $\wedge(\forall x, y)[x, y \in X \wedge x \lhd^+ y \rightarrow (\forall z)[x \lhd^+ z \rightarrow (z \lhd^* y \vee y \lhd^* z)]]$

Hence, simply postulating axiom (20) for all MSOL-based LFG grammars will guarantee decidability of the parsing problem for the new description format.

2.4 Equivalence with LFG expressiveness

Based on the hybrid specification of c-structure (using MSOL) and f-structure (using standard LFG feature descriptions) discussed in the previous section, we can convince ourselves pretty easily that the MSOL-based variant of the LFG formalism is neither more nor less expressive than standard LFG, i.e., the following equivalence holds:

(21) Class of standard LFG languages = class of MSOL-based LFG languages

For a given standard LFG grammar, we can construct an equivalent MSOL grammar, and vice versa. If we are given a standard LFG grammar, we can formulate MSOL formulae for each rewrite rule by directly describing the local tree configurations, using the immediate dominance relation $\lhd$ and precedence ($\prec$).[8] The f-annotations for each daughter category can be expressed directly, using

[8]Regular expressions on the right-hand side of LFG rules can also be captured fully by MSOL. For example, the Kleene star in the rewrite rule 'NP →DET ADJ* N' can be captured by saying:
$(\forall x)\mathrm{NP}(x) \rightarrow (\exists y_1 y_2 Y)[\mathrm{DET}(y_1) \wedge x \lhd y_1 \wedge \mathrm{N}(y_2) \wedge x \lhd y_2 \wedge (\forall z) z \in Y \rightarrow x \lhd z \wedge \mathrm{ADJ}(z) \wedge y_1 \prec z \prec y_2]$
(This formulation assumes that 'DET ADJ* N' is the only right-hand side for NP; if there were more possible right-hand sides, a disjunction over all of them would have to be used.)

f-annotation predicates which conform to (16) and (18), since the original annotations make only reference to ↑ and ↓, and there are only finitely many different f-annotations in a standard LFG grammar.

For a given MSOL-based grammar specification, a classical LFG can be obtained by first converting the MSOL description into an equivalent tree automaton (which can be done due to the equivalence discussed in section 2.2). The f-annotation predicates are treated as unanalyzed labels, similar to the category labels. The resulting tree automaton (compare (15) and figure 2) can be converted into a rewrite grammar by introducing a non-terminal category for each state and constructing the productions for the non-terminals by using all state transition terms that have the corresponding state as their target. Now, the f-annotation predicates can be converted into standard LFG f-annotations. The meta-constraint on f-annotation predicates (18) in MSOL-based LFG ensures that we can express the annotations in standard LFG.

Since in standard LFG and in MSOL-based LFG, the f-constraints are resolved in the same way, the equivalence at the rewrite rule level that we just sketched is all that is needed to show that equivalence (21) holds.

2.5 Discussion: Increased generality of descriptions

As the previous section showed, the expressive power of the LFG formalism is not increased by moving to MSOL-based c-structure desciptions (we have *descriptive equivalence*). So what is the benefit of the new format?

The answer is that we can now express principles generalizing over c-structure as explicit constraints that would otherwise have required a significantly blown-up representation. (Now, the compiled-out tree automata will contain the fine-grained distinctions, but these tree automata have a purely technical status; our theory is formulated at the level of MSOL tree descriptions.)[9]

This formalization expands the *description-based spirit of LFG* to c-structure specification (replacing a slightly derivational residual). With the hybrid system, the important computational properties of standard LFG carry over to the new system.

Computational considerations. Presently, the possibility of compiling out tree automata for an MSOL grammar formula is solely viewed as a theoretical tool, guaranteed by equivalences holding for the class of tree languages. But it may also be possible to devise computational procedures that perform the compilation for actual grammar specifications. There is a computational tool for such compilations, which was originally developed for hardware verification: the MONA system Klarlund and Moller (2001), Morawietz and Cornell (1999). As the experiments by Morawietz and Cornell show, the size of the tree automata gets extremely large when applied to non-trivial grammar formulae, at least in the heavily tree-based theoretical framework they were investigating. However, it may be possible to exploit properties of the special case of LFG grammar formulae to facilitate the compilation.

Potential further directions. While the focus on equivalence with standard LFG in the present paper motivated the use of a hybrid description system, it would be very interesting to explore a purely MSOL-based description of both c-structure and f-structure. A comparison with the hybrid approach may reveal interesting aspects of LFG and feature grammars in general.

[9]A further application of the generalized tree descriptions, besides in the formalization of theoretical principles holding on c-structure and the c-structure/f-structure mapping, might be treebank annotation principles as proposed by (Frank 2000). The framework discussed in the present paper could be used as a formal basis for Frank's generalized f-structure annotations.

3 An application: constraint specification in OT-LFG

Besides the application within classical LFG, the MSOL-based formulation of c-structure descriptions results in a significant clarification of the specification of computationally well-behaved OT-LFG systems. Kuhn's (2002, 2003) proof of the decidability of optimization for a suitably restricted OT-LFG system (based on Kaplan and Wedekind's (2000) context-freeness results) relies on OT constraints being "expressible local to a c-structure subtree". The relevant "locality" criterion is hard to pin down rigorously without a general tree description language: e.g., non-local dependencies are allowable as long as the information can be percolated through the tree using a finite set of category symbols. By demanding that all OT constraints are expressed in MSOL we get a very clear characterization of the "locality" criterion, and the context-free-grammar-guided decidability proof becomes much more perspicuous.

Section 3 of this paper attempts to present the computational issues addressed in the decidability proof of Kuhn (2002, 2003) in an accessible way and sketches the application of the logic-based c-structure specification proposed in the previous section in the context of this proof (compare Kuhn (in preparation)).

3.1 A brief introduction to OT-LFG

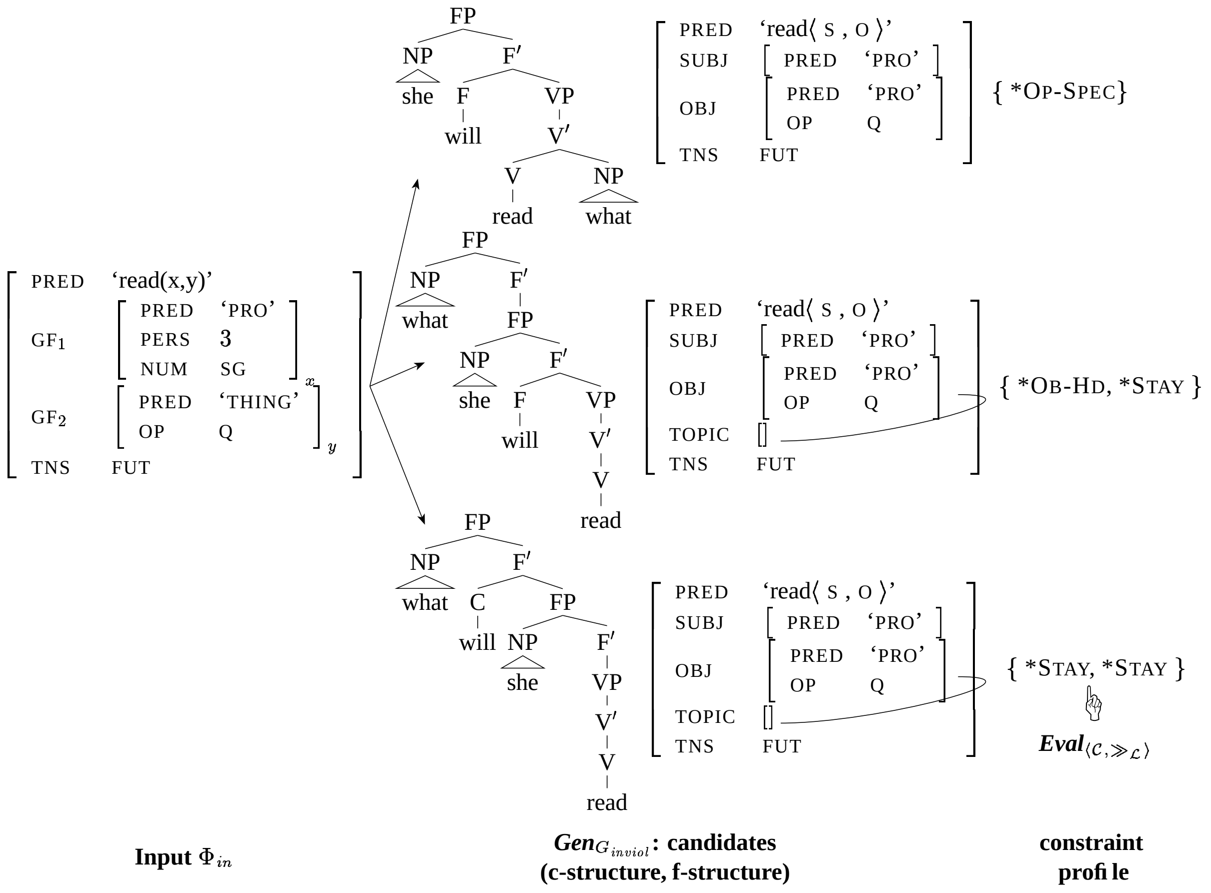

PRED 'read(x,y)'
GF1 PRED 'PRO' PERS 3 NUM SG x
GF2 PRED 'THING' OP Q y
TNS FUT
FP NP she F' F will VP V' V read NP what
PRED 'read⟨ S , O ⟩' SUBJ PRED 'PRO' OBJ PRED 'PRO' OP Q TNS FUT
{ *OP-SPEC}
FP NP what F' FP NP she F' F will VP V' V read
PRED 'read⟨ S , O ⟩' SUBJ PRED 'PRO' OBJ PRED 'PRO' OP Q TOPIC [] TNS FUT
{ *OB-HD, *STAY }
FP NP what F' C will FP NP she F' VP V' V read
PRED 'read⟨ S , O ⟩' SUBJ PRED 'PRO' OBJ PRED 'PRO' OP Q TOPIC [] TNS FUT
{ *STAY, *STAY }
Eval⟨C,≫C⟩
Input Φin
GenGinviol: candidates (c-structure, f-structure)
constraint profile

Figure 3: *The OT-LFG architecture: a graphical overview*

Figure 3 provides a graphical overview of OT-LFG as originally proposed by Joan Bresnan (e.g., in

Bresnan 2000) and discussed in detail in (Kuhn 2003). A general underlying LFG-style grammar defines the set of possible candidate representations. For a given input (a partially specified f-structure), the actual candidate set is defined as those c-structure/f-structure pairs for whose f-structure is subsumed by the input structure, without adding any semantic information.

The competing candidates are compared in terms of the violations of OT constraints that they incur. A (violable) OT constraint is a description of a subconfiguration (c-structure/f-structure, or a combination of both) of the candidate representations.

In figure 3, the constraint violation for each candidate are shown as a multiset of violation marks for the constraints OP-SPEC, OB-HD and STAY.[10] The language-specific ranking of the constraints determines which of the candidates is the most harmonic realization, making it the predicted grammatical realization of the underlying input.

3.2 Undecidability for unrestricted OT and the basis of a decidable system

The set-up of a syntactic OT system just sketched makes it possible to have an infinite number of competing candidates in a specific candidate set. As we will see, this is not in itself an unsolvable computational problem, but it *can* lead to a problem if the system is not otherwise restricted in a suitable way. We discuss the problem in the following; the solution of (Kuhn 2003), which is also assumed here, will be to allow for infinite candidate sets, but demand that constraints are not arbitrarily powerful.

The schematic example in figure 4 illustrates the computational problem that infinite candidate sets may pose without further restrictions on the formalism. (For a formally precise discussion of the issue, see Kuhn 2003, ch. 4.) Let us assume (i) that we have an underlying candidate generation grammar that licenses recursive structures like the Y-trees in figure 4. Moreover (ii), there is some high-ranking constraint (C_1) which we find to be violated in "small" structures, participating in the recursive construction.

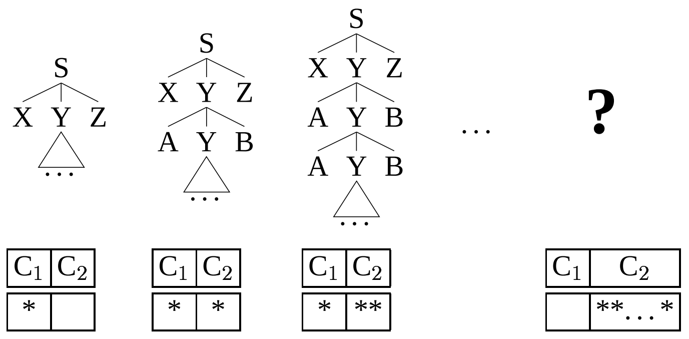


Figure 4: *Schematic depiction of the intuitive problem with infinite candidate sets*

Now we have to decide whether or not there is a way of avoiding the violation of C_1 – one possibility is to look at larger recursive structures. The constraint violation might be triggered by the "small" structures and could possibly go away. Let us furthermore say that the violation does not go

[10]Grimshaw's definitions for these constraints are the following (compare Grimshaw 1997, 374):

OP-SPEC	Syntactic operators must be in specifier position.
OB-HD	(Obligatory Head) A projection has a head.
STAY	Trace is not allowed.

For their translation into the OT-LFG framework, see (Bresnan 2000) and (Kuhn 2003, ch. 4).

away within the first n larger candidates we look at, using the recursion. Now the issue is: if there are infinitely many larger structures using the recursion, can we ever be sure that the C_1 violation *cannot* be avoided?

One may ask whether this type of problem is of any linguistic relevance for Optimality-Theoretic syntax. Maybe we could restrict the set of allowable candidate generation grammars, excluding recursion without a reflex in f-structure (which would make the problem disappear)? However as I will argue in the following subsection, this is not a viable option if we are interested in providing a formalization of the core idea of the Optimality-Theoretic approach, which is to derive cross-linguistic grammatical differences as an effect of constraint reranking. There is linguistic support for infinite candidate set.

3.2.1 Linguistic motivation for infinite candidate sets

Obviously, there can be no direct empirical support for the need of an infinite candidate set: there will always be just a finite set of winners (typically just a single winner). However, if we find a systematic OT explanation that relies on ranked constraints operating on a subset of candidates which are related to each other since they include more or fewer instances of otherwise unlimited recursion (which has no f-structure reflex), this will be a good indication that infinite candidate sets should not be excluded *a priori*.

The use of stacked functional projections in the extended projection architecture of Grimshaw (1997) is such an example. On top of the lexical VP projection, an arbitrary number of stacked FP projections is allowed (two of which will correspond to the classical IP and CP projections). If we translate this to an LFG setting, all of the functional categories will act as f-structure co-heads (following Bresnan 2001, ch. 6-7), so there is no f-structural nesting corresponding to the stacking in c-structure. This means that we will indeed get infinitely many candidates for any candidate set involving a lexical verb, as indicated in figure 5.

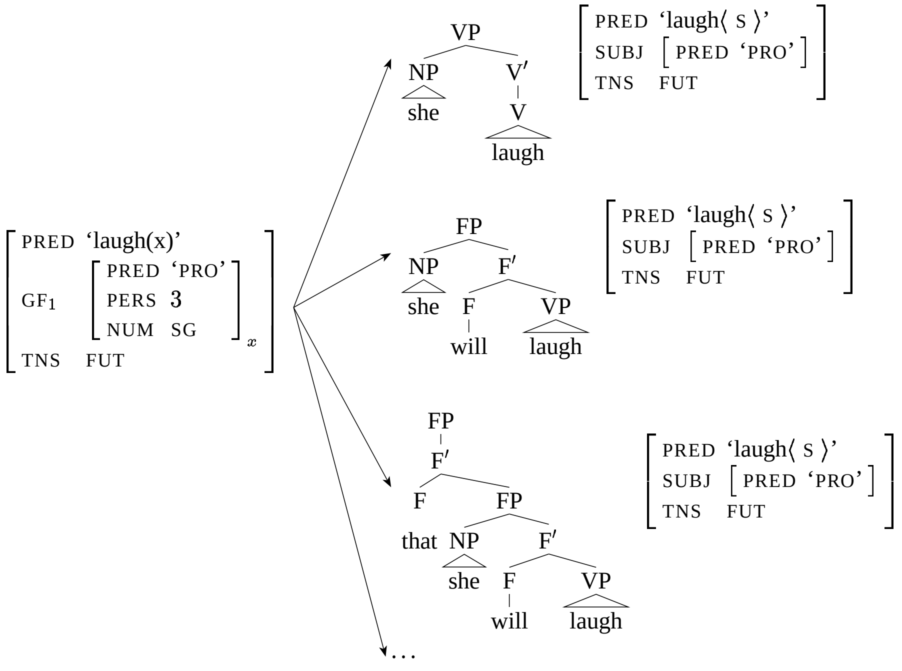

PRED 'laugh(x)'
GF1 [PRED 'PRO' PERS 3 NUM SG]x
TNS FUT
VP NP she V′ V laugh
PRED 'laugh⟨ S ⟩' SUBJ [PRED 'PRO'] TNS FUT
FP NP she F′ F will VP laugh
PRED 'laugh⟨ S ⟩' SUBJ [PRED 'PRO'] TNS FUT
FP F′ F that FP NP she F′ F will VP laugh
PRED 'laugh⟨ S ⟩' SUBJ [PRED 'PRO'] TNS FUT
...

Figure 5: *Infinite candidate set for a given f-structure due to stacking of functional projections*

In Grimshaw's analysis, the assumption of not just zero, one, or two, but arbitrarily many functional projections is justified by the derivation of English data with embedded clauses including an operator (compare Grimshaw 1997, 399ff). Constraint interaction leads to a winner (for English) which actually includes three functional projections. This indicates that a prior stipulation of a maximal depth for functional projection stacking (while technically still possible, using 3 or 4 as the limit) goes against the explanatory impact of the OT approach.

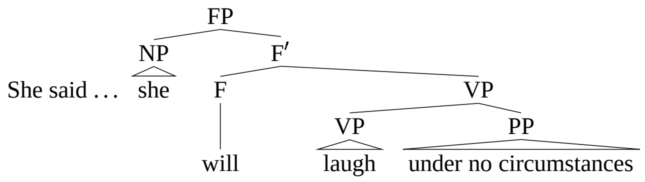
Structure (22) shows what would be the LFG correspondent of the input that Grimshaw (1997, 399ff) assumes; (23) presents a sequence of candidates for this input (only the c-structure is shown). In general, we should think of all candidates in OT-LFG as arising simultaneously in an abstract competition that does not involve the actual generation of the candidates in a cognitively real device. However, for pedagogical reasons this particular example presents the candidates in a sequential way, along with the tableau of candidates considered "so far". The sequence of presenting the candidates follows the order of increasing recursive stacking, suggestive of the abstract example given in figure 4. This should suggest that it is not a computational possibility to do the candidate generation first (for an infinite set of candidates!) and check the constraint violations afterwards; a concrete algorithm will have to interleave the two abstract processes of (i) candidate generation and (ii) constraint marking/harmony evaluation. For more details on example (22)/(23), see (Kuhn 2003, 185ff).[11]

(22) *Input*

$$\left[\begin{array}{ll}
\text{PRED} & \text{'say(x,y)'} \\
\text{GF}_1 & \left[\begin{array}{ll} \text{PRED} & \text{'PRO'} \\ \text{PERS} & 3 \\ \text{NUM} & \text{SG} \end{array}\right]_x \\
\text{GF}_2 & \left[\begin{array}{ll}
\text{PRED} & \text{'laugh(x)'} \\
\text{GF}_1 & \left[\begin{array}{ll} \text{PRED} & \text{'PRO'} \\ \text{PERS} & 3 \\ \text{NUM} & \text{SG} \end{array}\right]_x \\
\text{MOD} & \left[\begin{array}{ll} \text{PRED} & \text{'U. N. C.'} \\ \text{OP} & + \end{array}\right] \\
\text{TNS} & \text{FUT}
\end{array}\right]_y
\end{array}\right]$$

(23) a.

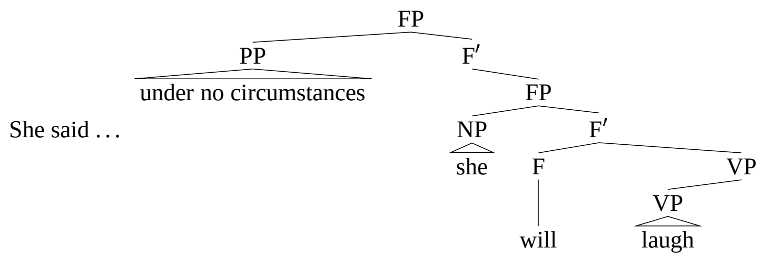
	PURE-EP	OP-SPEC	OB-HD	STAY
[$_{IP}$ she will [$_{VP}$ laugh u.n.c.]]		*		



b.

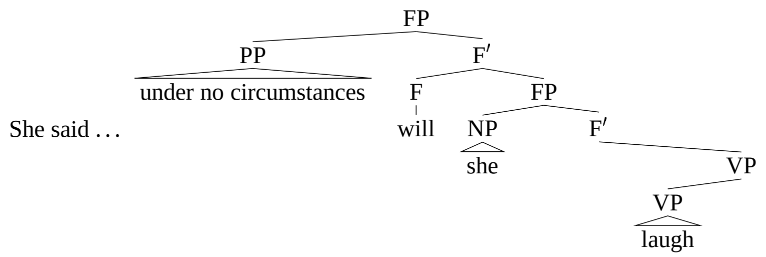
	PURE-EP	OP-SPEC	OB-HD	STAY
[$_{IP}$ she will [$_{VP}$ laugh u.n.c.]]		*		
[$_{XP}$ u.n.c. [$_{IP}$ she will [$_{VP}$]]	*		*	*

[11]The PURE-EP constraint is defined as follows (Grimshaw 1997, 394): "No adjunction takes places to the highest node in a subordinate extended projection; and no movement takes place into the highest head of a subordinate extended projection." For the other constraints, compare footnote 10.



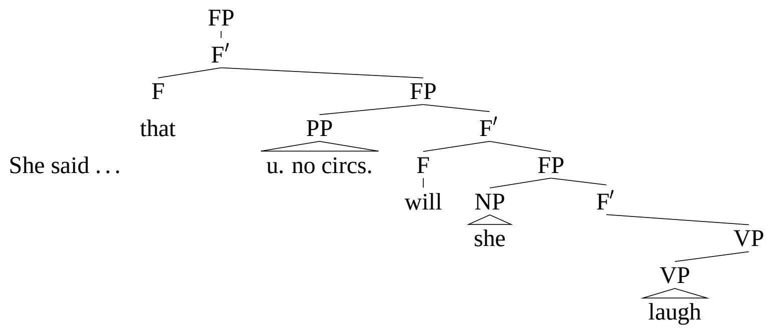
c.

	PURE-EP	OP-SPEC	OB-HD	STAY
[$_{\text{IP}}$ she will [$_{\text{VP}}$ laugh u.n.c.]]		*		
[$_{\text{XP}}$ u.n.c. [$_{\text{IP}}$ she will [$_{\text{VP}}$]]	*		*	*
[$_{\text{XP}}$ u.n.c. will [$_{\text{IP}}$ she [$_{\text{VP}}$]]	*			**



d.

	PURE-EP	OP-SPEC	OB-HD	STAY
[$_{\text{IP}}$ she will [$_{\text{VP}}$ laugh u.n.c.]]		*		
[$_{\text{XP}}$ u.n.c. [$_{\text{IP}}$ she will [$_{\text{VP}}$]]	*		*	*
[$_{\text{XP}}$ u.n.c. will [$_{\text{IP}}$ she [$_{\text{VP}}$]]	*			**
☞[$_{\text{CP}}$ that [$_{\text{XP}}$ u.n.c. will [$_{\text{IP}}$ she [$_{\text{VP}}$]]]]				**



Note that the ultimate winner with its three levels of stacked FPs avoids all violations of high-ranking constraints, in contrast to the "smaller" candidates candidates considered "earlier on". The winning candidate only violates the low-ranking STAY constraint.

3.2.2 Decidability despite infinity of the candidate set

The example in the previous subsection showed that from the point of view of linguistic modeling, infi nite candidate sets should not be excluded *a priori*. As it turns out, infi nity of the candidate set is not a problem by itself. As long as there are fi nitely many equivalence classes of candidates to deal with, the problem will be manageable. (Kuhn 2003) presents a decidability proof for OT generation based on a constraint concept in which all constraints are anchored in a local tree configuration. This makes it possible to discard infinitely large equivalent classes of candidates as impossible winners and leaves only a finite set for comparison.

The construction is based on the following result of Kaplan and Wedekind (2000): Generation from a given f-structure (within classical LFG) produces a context-free language. Kaplan and Wedekind present a construction that exploits the strict f-structural constraints imposed by the goal of generating from a particular f-structure. These constraints are folded into the c-structure symbols, creating a large but finite set of category symbols that take over the role of all f-structural restrictions in a normal LFG grammar (for the particular given f-structure for which strings are generated). Ultimately, no f-annotations are needed anymore, and we get a context-free grammar that generates exactly those strings which the original LFG grammar accepted for the given f-structure.

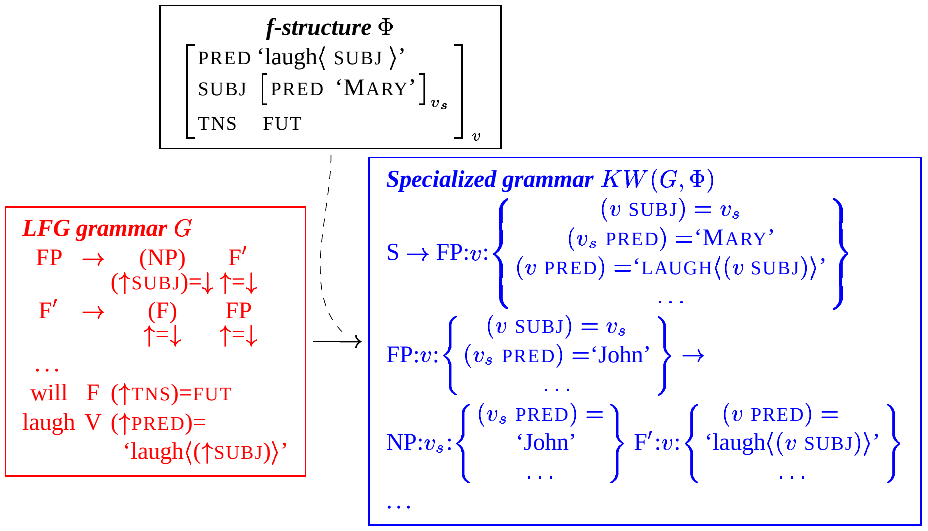

f-structure Φ
PRED 'laugh⟨ SUBJ ⟩'
SUBJ [PRED 'MARY'] v_s
TNS FUT
v
LFG grammar G
FP → (NP) F′
(↑SUBJ)=↓ ↑=↓
F′ → (F) FP
↑=↓ ↑=↓
. . .
will F (↑TNS)=FUT
laugh V (↑PRED)= 'laugh⟨(↑SUBJ)⟩'
Specialized grammar $KW(G,\Phi)$
S → FP:v:{ (v SUBJ) = v_s, (v_s PRED) ='MARY', (v PRED) ='LAUGH⟨(v SUBJ)⟩', . . . }
FP:v:{ (v SUBJ) = v_s, (v_s PRED) ='John', . . . } →
NP:v_s:{ (v_s PRED) = 'John', . . . } F′:v:{ (v PRED) = 'laugh⟨(v SUBJ)⟩', . . . }
. . .

Figure 6: *Illustration of the construction of Kaplan and Wedekind (2000)*

Figure 6 provides a schematic illustration of Kaplan and Wedekind's construction (see their paper or (Kuhn 2003, ch. 6) for details). Given an LFG grammar G and an f-structure Φ, a specialized context-free grammar is constructed which we may call $KW(G,\Phi)$. The augmented c-structure categories in this grammar have the form XP:$v_{x_1 \ldots x_n}$:Γ. Γ encodes a set of instantiated f-constraints percolated bottom-up.[12] The consistency of all the f-constraints that are percolated up to the root symbol (and Completeness/Coherence) is checked by a selective introduction of productions for a new start symbol. (This is feasible since the existence of a fixed Φ guarantees that there is a finite set of possible f-constraint instantiations for the grammar.)

Kuhn (2000, 2003) exploits the "KW construction" for showing that generation-based optimization in OT-LFG is decidable, based on two assumptions:

1. Each candidate's f-structure is identical to the OT input (modulo addition of a bounded amount of information);

2. All OT constraints can be anchored local to a c-structure category.

[12]The expressions following the first colon – v, v_s – are used to fix the instantiation of the metavariables ↑, ↓ to specific paths taken from Φ (such as the empty path, or the path SUBJ, or COMP SUBJ etc., in large f-structures); all possible combinations of such fixed instantiations are created.

For the locality restriction, there has not been a fully satisfactory formulation so far. The logic-based formulation of tree constraints brought forward in section 2 of the present paper will actually fi ll this gap.

Technically, Kuhn's (2003) application of the KW construction for OT-LFG involves an additional step of transforming the LFG grammar used for candidate generation (G_{inviol}) into a version that includes explicit markings of locally incurred constraint violations as part of an expanded c-structure category format. We may refer to the result of transforming a grammar in this way as $O_{\mathcal{C}}(G)$.

When the KW construction is applied to $O_{\mathcal{C}}(G_{inviol})$ for a candidate generation LFG grammar G_{inviol} and an underlying input f-structure Φ_{in}, we get a context-free grammar $KW(O_{\mathcal{C}}(G_{inviol}), \Phi_{in})$ (this is illustrated schematically in fi gure 7). This resulting context-free grammar may still include recursive rules (thus generating infi nitely many strings). But the structures created by traversing a chain of recursive rules are at most as harmonic as a smaller candidate already generated (this is a consequence of having encoded all local constraint violations in the category symbols). So there is an effective way of generating the full set of winners. (In well-behaved OT systems this will be a small fi nite set, but even if there are infi nitely many candidates for the optimal constraint profi le, we get a context-free grammar that produces exactly those winners.)

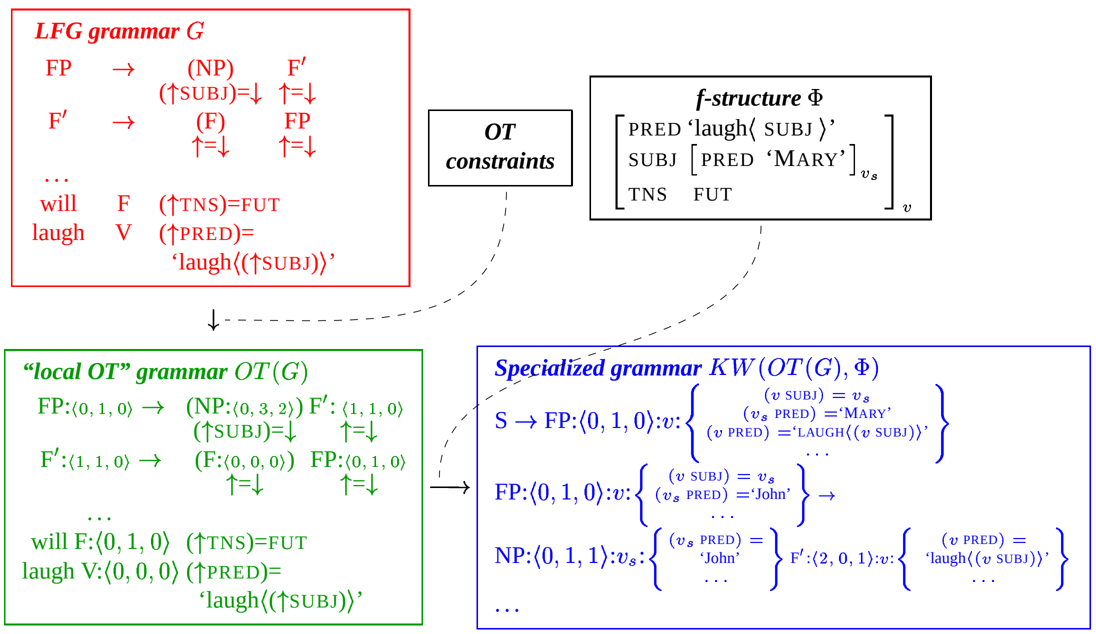

LFG grammar G
FP → (NP) (↑SUBJ)=↓ F′ ↑=↓
F′ → (F) ↑=↓ FP ↑=↓
...
will F (↑TNS)=FUT
laugh V (↑PRED)= 'laugh⟨(↑SUBJ)⟩'
OT constraints
f-structure Φ
PRED 'laugh⟨ SUBJ ⟩'
SUBJ [PRED 'MARY'] v_s
TNS FUT
v
"local OT" grammar OT(G)
FP:⟨0,1,0⟩ → (NP:⟨0,3,2⟩) (↑SUBJ)=↓ F′:⟨1,1,0⟩ ↑=↓
F′:⟨1,1,0⟩ → (F:⟨0,0,0⟩) ↑=↓ FP:⟨0,1,0⟩ ↑=↓
...
will F:⟨0,1,0⟩ (↑TNS)=FUT
laugh V:⟨0,0,0⟩ (↑PRED)= 'laugh⟨(↑SUBJ)⟩'
Specialized grammar KW(OT(G), Φ)
S → FP:⟨0,1,0⟩:v:{ (v SUBJ) = v_s (v_s PRED) ='MARY' (v PRED) ='LAUGH⟨(v SUBJ)⟩' ... }
FP:⟨0,1,0⟩:v:{ (v SUBJ) = v_s (v_s PRED) ='John' ... } →
NP:⟨0,1,1⟩:v_s:{ (v_s PRED) = 'John' ... } F′:⟨2,0,1⟩:v:{ (v PRED) = 'laugh⟨(v SUBJ)⟩' ... }
...

Figure 7: *Illustration of the decidability construction for OT-LFG generation*

Summing up the result of (Kuhn 2003, ch. 6), the specialized grammar $KW(OT(G), \Phi)$ is guaranteed to generate the optimal OT candidates in its non-recursive part. This guarantees decidability of generation-based optimization.[13]

3.3 The constraint locality condition

As was pointed out above, the constraint locality condition assumed in the decidability proof of (Kuhn 2003) has had no fully satisfactory formulation so far. Under the assumption that OT constraints have

[13]It does not guarantee decidability of the recognition/parsing problem for expressive optimization systems. This requires further assumptions about contextual recoverability or a bidirectional optimization scheme (Kuhn 2003, sec. 6.3).

one of the implicative forms in (24), the generation construction sketched in the previous section can be formulated.

(24) a.

$$\begin{array}{ccc} N & \Rightarrow & N' \\ S & & S' \end{array}$$

where N, N' are descriptions of nonterminals of G_{inviol}; S, S' are standard LFG f-annotations of constraining equations with $\uparrow$ as the only f-structure metavariable.

b.

$$\begin{array}{ccc} \overbrace{\begin{array}{ccc} \rho & M & \sigma \\ & S & \end{array}}^{N} & \Rightarrow & \overbrace{\begin{array}{ccc} \rho' & M' & \sigma' \\ & S' & \end{array}}^{N'} \end{array}$$

where N, N', M, M' are descriptions of nonterminals of G_{inviol}; N, N' refer to the mother in a local subtree configuration, M, M' refer to the same daughter category; $\rho, \rho', \sigma, \sigma'$ are regular expressions over nonterminals; S, S' are standard f-annotations as in (24a).

However, for non-trivial c-structural OT constraints, a conversion of the c-structure representation had to be assumed, prior to formulating the constraints. An example from (Kuhn 2003, 97) is given in (25).

(25) IPccxhy vs. IPccxhn: *c-commanding extended head yes/no*

C′	→	C	IPccxhy
C′	→		IPccxhn
IPccxhy	→	(XP)	I′ccxhy
IPccxhn	→	(XP)	I′ccxhn
I′ccxhn	→	(I)	(VP)
I′ccxhy	→	(I)	(VP)

As has been noted, even non-local conditions can be encoded, using a GPSG-style slash feature as part of the c-structure categories. But at the same time this reveals a certain problem: it is not entirely transparent what the actual limitations are that are imposed by the constraint locality condition.

Although reformulations of the c-structure rules along the lines of (25) could be devised for a given construction and given OT constraints, there has been no mechanical way of getting from the constraint formulation to the required representation.

3.4 Using a logic-based formulation of constraints

A (MSO) logic-based formulation of OT constraints resolves the issue addressed at the end of the previous section. OT constraints can be formalized as MSOL formulae with a free variable (ranging over tree nodes). F-structure constraints can be included, using the hybrid specification technique discussed in section 2.3.[14] (26) is an example of an MSOL-based formulation of the constraint OB-HD, making use of the *ExtHd* predicate defined above and a predicate *XBar0* that can be defined in an obvious way.

(26) OB-HD (Bresnan 2000, (21))

Every projected category has a lexically filled [extended, JK] head.

$\mathit{ObHd}(x) \equiv \neg \mathit{XBar0}(x) \rightarrow [(\exists y)[\mathit{ExtHd}(y, x)]]$

[14]OT constraints addressing f-structure may include only constraining equations.

For the decidability construction discussed in section 3.2.2, the full cross-product of local constraint (non-)violations can now be expressed by a disjunction of such open formulae $C_i(x)$; we can use this disjunction to introduce "constraint violation label" predicates $\langle n_1, n_2, \ldots, n_k\rangle(x)$, where $n_i \in \{0, 1\}$ (multiple violations of the same constraint will always originate from different nodes).

(27) $$\begin{array}{l} (C_1(x) \wedge C_2(x) \wedge \ldots C_k(x)) \leftrightarrow \langle 0, 0, \ldots, 0\rangle(x) \vee \\ (\neg C_1(x) \wedge C_2(x) \wedge \ldots C_k(x)) \leftrightarrow \langle 1, 0, \ldots, 0\rangle(x) \vee \\ (\neg C_1(x) \wedge \neg C_2(x) \wedge \ldots C_k(x)) \leftrightarrow \langle 1, 1, \ldots, 0\rangle(x) \vee \\ \vdots \\ (\neg C_1(x) \wedge \neg C_2(x) \wedge \ldots \neg C_k(x)) \leftrightarrow \langle 1, 1, \ldots, 1\rangle(x) \end{array}$$

The constraint violation labels defined in this way can be used directly to control the generation of "non-recursive" trees in the KW-construction step.

So, we can conclude that the MSOL-based constraint formulation clarifies the formal preconditions for decidability of optimization to a great extent: Any constraint expressible as an open formula $C_i(x)$ of MSOL will be usable in an OT-LFG system without jeopardizing decidability of OT generation. For nonlocal dependencies it is no longer necessary to manually construct a c-structure level representation like the one in (25) that takes care of the step-by-step relationship. The theoretical results on the relation between MSOL and tree automata (and thus the tree skeleton of context-free string languages) ensure that there will be a way of compiling out the grammar.[15]

4 Conclusion

In the first part of this paper (section 2), I argued that MSOL-based tree descriptions make it possible to express theoretical principles generalizing over c-structure in a direct descriptive way (which is not possible under the standard LFG characterization of c-structure). If the relation between c-structure and f-structure is realized by a hybrid description scheme (inheriting the standard LFG feature logic for f-structural constraint resolution), descriptive equivalence of standard LFG and the new c-description format follows in a fairly straightforward way. As the second part of the paper illustrated (section 3), open formulae in MSOL provide an elegant way of characterizing the locality condition of OT-LFG constraints, which is required to guarantee decidability of generation-based optimization with infinite candidate sets. A more thorough discussion is provided in Kuhn (in preparation).

References

Bresnan, Joan. 2000. Optimal syntax. In Joost Dekkers, Frank van der Leeuw, and Jeroen van de Weijer (eds.), *Optimality Theory: Phonology, Syntax, and Acquisition*. Oxford University Press.

Bresnan, Joan. 2001. *Lexical-Functional Syntax*. Oxford: Blackwell.

Butt, Miriam, Stefanie Dipper, Anette Frank, and Tracy Holloway King. 1999a. Writing large-scale parallel grammars for english, french, and german. In M. Butt and T. H. King (eds.), *Proceedings of the LFG99 Conference, Manchester, UK*, CSLI Proceedings Online.

Butt, Miriam, Martin Forst, Tracy H. King, and Jonas Kuhn. 2003. The feature space in parallel grammar writing. In *Proceedings of ESSLLI'03-Workshop on Ideas and Strategies in Multilingual Grammar Development, August 2003, Vienna*.

[15]Just like alluded to in section 2.5, in the OT-LFG context too computational tools like MONA (Klarlund and Moller 2001, Morawietz and Cornell 1999) may be used to perform this compilation.

Butt, Miriam, Tracy King, Maria-Eugenia Niño, and Fr´ed´erique Segond. 1999b. *A Grammar Writer's Cookbook*. Number 95 in CSLI Lecture Notes. Stanford, CA: CSLI Publications.

Clement, Lionel, and Alexandra Kinyon. 2003. Generating LFGs with a MetaGrammar. In *Proceedings of the LFG 2003 Conference, Saratoga Springs, NY, USA*. to appear.

Frank, Anette. 2000. Automatic f-structure annotation of treebank trees. In M. Butt and T. H. King (eds.), *Proceedings of the LFG 2000 Conference, Berkeley, CA*, CSLI Proceedings Online.

G´ecseg, Ferenc, and Magnus Steinby. 1997. Tree languages. In Grzegorz Rozenberg and Arto Salomaa (eds.), *Handbook of Formal Languages*, pp. 1–68. Berlin/Heidelberg: Springer Verlag.

Grimshaw, Jane. 1997. Projection, heads, and optimality. *Linguistic Inquiry* 28:373–422.

Kaplan, Ronald M., and Joan W. Bresnan. 1982. Lexical-Functional Grammar: a formal system for grammatical representation. In Joan W. Bresnan (ed.), *The Mental Representation of Grammatical Relations*, chapter 4, pp. 173–281. Cambridge, MA: MIT Press.

Kaplan, Ronald M., and Jürgen Wedekind. 2000. LFG generation produces context-free languages. In *Proceedings of COLING-2000*, pp. 297–302, Saarbrücken.

Klarlund, Nils, and Anders Moller. 2001. Mona version 1.4 user manual. Technical report, BRICS, University of Aarhus.

Kuhn, Jonas. 1999a. Meta-descriptions of rules for generalization in constraint-based grammar design. Ms. Institut für maschinelle Sprachverarbeitung, Universität Stuttgart.

Kuhn, Jonas. 1999b. Towards a simple architecture for the structure-function mapping. In M. Butt and T. H. King (eds.), *Proceedings of the LFG99 Conference, Manchester, UK*, CSLI Proceedings Online.

Kuhn, Jonas. 2000. Processing Optimality-theoretic syntax by interleaved chart parsing and generation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, pp. 360–367, Hongkong.

Kuhn, Jonas. 2002. OT syntax – decidability of generation-based optimization. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02), Philadelphia*, pp. 48–55.

Kuhn, Jonas. 2003. *Optimality-Theoretic Syntax—A Declarative Approach*. Stanford, CA: CSLI Publications.

Kuhn, Jonas. in preparation. Candidate generation in Optimality-Theoretic Syntax—conditions for decidability [working title]. Ms. The University of Texas at Austin.

Morawietz, Frank, and Tom Cornell. 1999. The MSO logic-automaton connection in linguistics. In Alain Lecomte, François Lamarche, and Guy Perrier (eds.), *Logical Aspects of Computational Linguistics, Second International Conference, LACL '97, Nancy, France, September 22-24, 1997, Selected Papers*, volume 1582 of *Lecture Notes in Computer Science*. Springer.

Muskens, Reinhard. 2001. Categorial grammar and lexical-functional grammar. In M. Butt and T. H. King (eds.), *Proceedings of the LFG 2001 Conference, University of Hong Kong*, CSLI Proceedings Online.

Pollard, Carl J., and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. Chicago, London: University of Chicago Press.

Rogers, James. 1997. "Grammarless" Phrase Structure Grammar. Ms., Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia, PA.

Rogers, James. 1998. *A Descriptive Approach to Language-Theoretic Complexity*. Stanford, CA: CSLI Publications.

Rogers, James. 2001. wMSO theories as grammar formalisms. Ms., Earlham College, Richmond, Indiana.

Sells, Peter. 2001. *Structure, Alignment and Optimality in Swedish*. Stanford: CSLI Publications.

Jonas Kuhn
`jonask@mail.utexas.edu`
Department of Linguistics
1 University Station, B5100
University of Texas at Austin
Austin, TX 78712-1196
USA