Lexicalized Syntax and Topology for Non-projective Dependency Grammar

Denys Duchier [1]

Programming Systems Lab
Universität des Saarlandes
Saarbrücken, Germany

Abstract

We propose a lexicalized formulation of dependency grammar that addresses both immediate dependence and linear precedence. Our approach distinguishes two orthogonal, yet mutually constraining dependency trees: an ID tree of syntactic dependencies and a LP tree of topological dependencies. The ID tree is non-ordered, non-projective and its edges are labeled by grammatical functions. The LP tree is ordered and projective and expresses licensed linearizations; its edges are labeled by topological fields. The LP tree can be regarded as deriving from the ID tree through a process of emancipation controlled by lexicalized constraints and principles. In the present article, we formalize valid ID/LP analyses and show how they can be characterized as the solutions of a constraint satisfaction problem. The latter can be solved by constraint programming and forms the basis of our implementation.

1 Introduction

We propose a lexicalized formulation of dependency grammar which extends the non-projective account of syntax of [3] with an account of linear precedence inspired by the classical model of topological fields [1]. In this framework, an analysis consists of two mutually constraining trees: a tree of syntactic dependencies (ID tree) and a tree of topological dependencies (LP tree). The ID tree is non-ordered and non-projective, and its edges are labeled by syntactic relations. The LP tree is partially ordered and projective, and its edges are labeled by topological fields. The shape of the LP tree is a flattening of the ID tree's obtained by an emancipation process allowing nodes to 'climb up'. Our theory is formulated in term of lexicalized constraints and of principles governing climbing conditions. In [4], we described its application to an account of word-order phenomena in the German verb complex. In the present article,

[1] Email: duchier@ps.uni-sb.de

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

we focus on its formalization and state a formal well-formedness condition that precisely characterizes the valid analyses. Furthermore, this condition can also be interpreted as a constraint program and forms the basis of our implementation.

2 Dependency Trees for Syntax and Topology

Consider the sentence:

(dass) Maria einen Mann wird lieben können
(that) $Maria_{nom}$ $a\ man_{acc}$ *will love can*

The corresponding syntax tree (ID tree) is shown in Figure 1. This tree is unordered and non-projective. Its edges are labeled by grammatical functions such as subject or object.

We associate with it a topological tree (LP tree) ordered and projective, which is formed from the same set of nodes, but different edges. The edges of the LP tree are labeled by topological fields. One ID tree may give rise to several LP trees: Figure 2 displays 3 possibilities for the ID tree of Figure 1.

The edge labels of the LP tree are called *(external) fields* and are totally ordered: $\mathsf{df} \prec \mathsf{mf} \prec \mathsf{vc} \prec \mathsf{xf}$. This induces a linear precedence among the daughters of a node. This precedence is partial because daughters with the same field label may be freely permuted (which is the basis of our account of scrambling in the Mittelfeld).

In order to fully linearize a LP tree, each node must also be positioned with respect to its daughters. We achieve this by additionally assigning a label (d, n, or v) to each node. These are called *internal* fields; in Figure 2 they are shown on the vertical dotted lines joining a node to the word it stands for in the sentence. It is the combined set of internal and external fields which is totally ordered: $\mathsf{df} \prec \mathsf{d} \prec \mathsf{n} \prec \mathsf{mf} \prec \mathsf{vc} \prec \mathsf{xf}$

For an edge $w\text{-}\ell\!\rightarrow\! w'$ in the ID tree, we say that w is the head of w'. For a similar edge in the LP tree, we say that w is the host of w' or that w' lands on w. The shape of the LP tree is a flattened version of the ID tree's obtained by an emancipation process allowing nodes to 'climb up' subject to the following principles:

Principle 1. A node must land on a transitive head,[2] i.e. its host in the LP tree must be an ancestor in the ID tree.

Principle 2. A node may not climb through a barrier, i.e. none of its ID ancestors up to (but not including) its host may *block* its emancipation. We consider only a simple version, where a node can block a set of syntactic relations.

Principle 3. A node must land on, or climb higher than its head. The intuition here is that when a node climbs, it takes its entire subtree along.

[2] This is Bröcker's terminology.

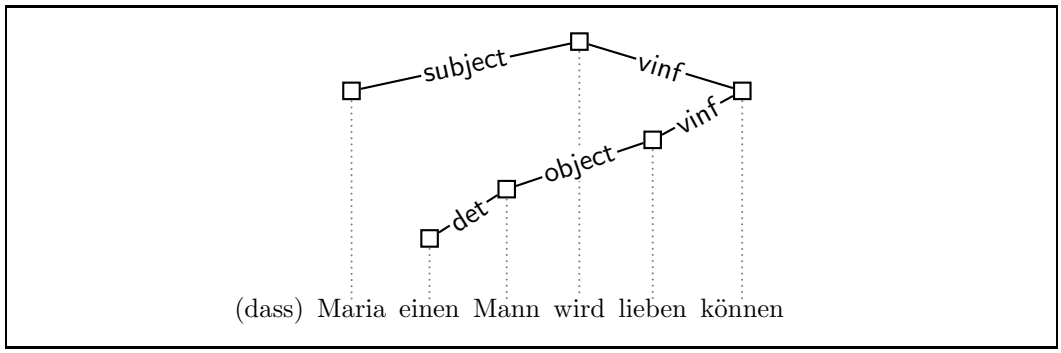

Fig. 1. Syntactic Dependency Tree

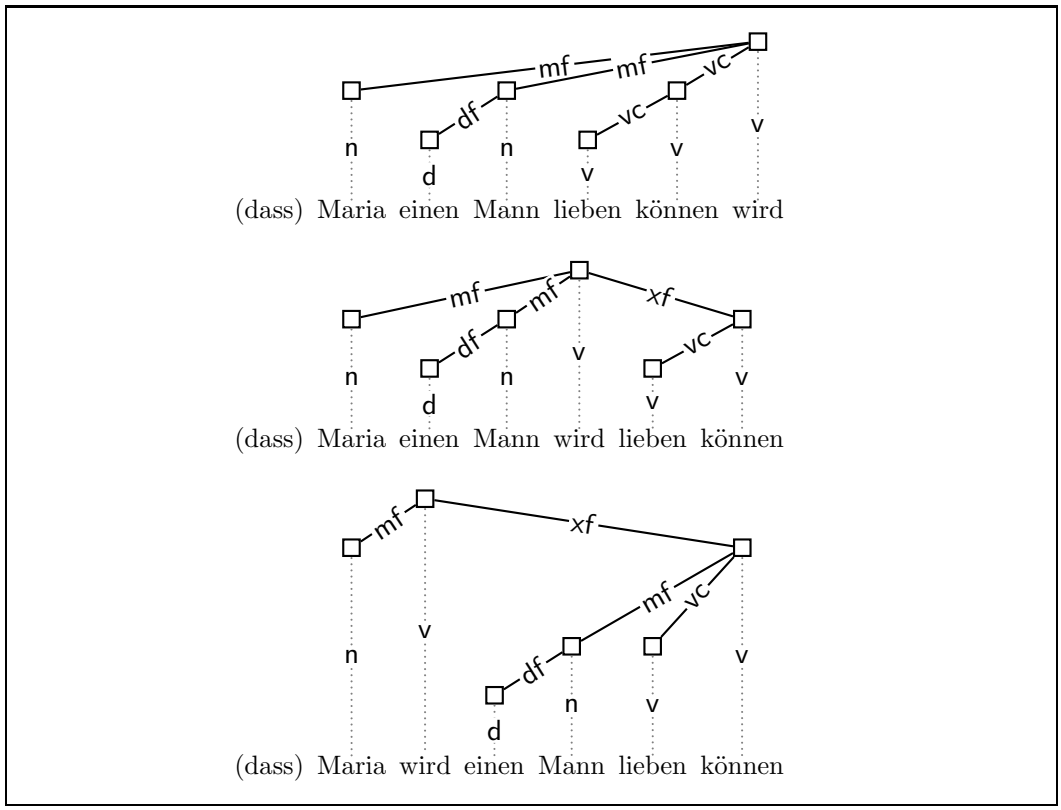

Fig. 2. Topological Dependency Trees

3 Labeled Trees

In this section we review the formalization of finite labeled trees presented in [3] and characterize the trees which can be formed from a finite set V of nodes and a finite set $\mathcal{L}$ of edge labels as the solutions of a constraint satisfaction problem.

We assume given an infinite set of node variables $\mathcal{V}$ and a finite set of edge labels $\mathcal{L}$. We write $\mathbf{G}(\mathcal{V}, \mathcal{L})$ for the set of finite graphs $G = (V, E)$ formed from a finite set of nodes $V \subseteq \mathcal{V}$ and the set of labeled edges $E \subseteq V \times V \times \mathcal{L}$. Note that, since we assume E to be a set, we only consider graphs without

$$
\begin{array}{ll}
\Phi_{\mathsf{ID}}(V,\mathcal{L}) \;\equiv & \\
\quad V = \mathsf{roots} \;\uplus\; \uplus\{\mathsf{daughters}(w) \mid w \in V\} & \\
\wedge \quad |\mathsf{roots}| = 1 & \\
\wedge\, \forall w \in V & \\
\quad \mathsf{eqdown}(w) = \{w\} \uplus \mathsf{down}(w) & \\
\wedge \quad \mathsf{down}(w) = \cup\{\mathsf{eqdown}(w') \mid w' \in \mathsf{daughters}(w)\} & \\
\wedge \quad \mathsf{equp}(w) = \{w\} \uplus \mathsf{up}(w) & \\
\wedge \quad \mathsf{up}(w) = \cup\{\mathsf{equp}(w') \mid w' \in \mathsf{mothers}(w)\} & \\
\wedge\, \mathsf{daughters}(w) = \uplus\{\ell(w) \mid \ell \in \mathcal{L}\} & \\
\wedge\, \mathsf{mothers}(w) \subseteq V \;\wedge\; |\mathsf{mothers}(w)| \leq 1 & \\
\wedge\, \forall w' \in V \quad w' \in \mathsf{daughters}(w) \;\equiv\; w \in \mathsf{mothers}(w') &
\end{array}
$$

Fig. 3. Well-formedness condition of labeled trees

duplicate edges. We write $\mathbf{G}(V,\mathcal{L})$ for the graphs in $\mathbf{G}(\mathcal{V},\mathcal{L})$ whose node set is V.

A finite graph G is a tree whenever it satisfies the following conditions: (a) each node has at most one incoming edge, (b) there is precisely one node (the root) with no incoming edge, (c) there are no cycles. We write $\mathbf{T}(\mathcal{V},\mathcal{L})$ for the subset of $\mathbf{G}(\mathcal{V},\mathcal{L})$ satisfying these conditions and $\mathbf{T}(V,\mathcal{L})$ for the trees in $\mathbf{T}(\mathcal{V},\mathcal{L})$ whose node set is V. We are going to formulate a condition $\Phi_{\mathsf{ID}}(V,\mathcal{L})$ which a finite graph $G = (V,E) \in \mathbf{G}(V,\mathcal{L})$ must satisfy in order to be in $\mathbf{T}(V,\mathcal{L})$.

We write $w\text{-}\ell\!\rightarrow\! w'$ for a directed labeled edge (w,w',ℓ) and $w\text{-}\ell\!\rightarrow_G\! w'$ for $w\text{-}\ell\!\rightarrow\! w' \in E$. We define the successor relation $\rightarrow_G = \cup\{\text{-}\ell\!\rightarrow_G \mid \ell \in \mathcal{L}\}$ and write $\rightarrow_G^+$ and $\rightarrow_G^*$ for its transitive and reflexive transitive closures. Given a relation $R \subseteq V \times V$, we define functions $R, R^{-1} : V \rightarrow 2^V$ and the overloading $R : 2^V \rightarrow 2^V$ as follows:

$$R(x) = \{y \mid (x,y) \in R\} \quad R^{-1}(y) = \{x \mid (x,y) \in R\} \quad R(S) = \cup\{R(x) \mid x \in S\}$$

In this manner, the edges of a labeled graph G induce the following functions:

$$
\begin{array}{rlrl}
\ell_G &= \text{-}\ell\!\rightarrow_G & \mathsf{down}_G &= \rightarrow_G^+ \\
\mathsf{daughters}_G &= \rightarrow_G & \mathsf{eqdown}_G &= \rightarrow_G^* \\
\mathsf{mothers}_G &= \mathsf{daughters}_G^{-1} & \mathsf{up}_G &= \mathsf{down}_G^{-1} \\
\mathsf{roots}_G &= V \setminus \rightarrow_G(V) & \mathsf{equp}_G &= \mathsf{eqdown}_G^{-1}
\end{array}
$$

Given these definitions and the treeness conditions (a), (b) and (c), we formulate, in Figure 3, a condition $\Phi_{\mathsf{ID}}(V,\mathcal{L})$ which a finite graph $G{=}(V,E) \in \mathbf{G}(\mathcal{V},\mathcal{L})$ must satisfy to be in $\mathbf{T}(V,\mathcal{L})$. This forms a constraint satisfaction problem (CSP) expressed in terms of variable roots of type 2^V and functional variables down, eqdown, up, equp, $\mathsf{mothers}$, $\mathsf{daughters}$ and ℓ (for all $\ell \in \mathcal{L}$) of type $V \rightarrow 2^V$. $\mathbf{T}(V,\mathcal{L})$ is in bijection with the solutions of $\Phi_{\mathsf{ID}}(V,\mathcal{L})$.

4 Configuration Lexicon

We now introduce the notion of a configuration lexicon and define the set of labeled trees which it licenses. A configuration lexicon $(\mathit{Lex}, \mathsf{valency}, \mathsf{labels})$ for $\mathbf{T}(V, \mathcal{L})$ consists of a finite set Lex of variables, called lexical entries, and two functions: $\mathsf{valency} : \mathit{Lex} \to \mathcal{L} \to 2^{\mathbb{N}}$, and $\mathsf{labels} : \mathit{Lex} \to 2^{\mathcal{L}}$.

An attributed tree is a triple (V, E, α) where $(V, E) \in \mathbf{T}(V, \mathcal{L})$ and $\alpha : V \to \mathit{Lex}$ assigns a lexical entry to each of its nodes. It is well-configured if it satisfies:

$$\forall w \in V,\ \forall \ell \in \mathcal{L} \qquad |\ell(w)| \in \mathsf{valency}(\alpha(w))(\ell) \tag{1}$$

$$\forall w, w' \in V,\ \forall \ell \in \mathcal{L} \qquad w\text{-}\ell\!\to\! w' \in E \ \Rightarrow\ \ell \in \mathsf{labels}(\alpha(w')) \tag{2}$$

$\mathsf{valency}(\alpha(w))(\ell)$ restricts the licensed number of w's out-going ℓ-edges. Therefore, $\mathsf{valency}(\alpha(w))$ represents a constraint on the edges *offered* by w. Conversely $\mathsf{labels}(\alpha(w'))$ is a restriction on the in-coming edges *accepted* by w'.

We write $\mathbf{T}(\mathcal{V}, \mathcal{L}, \mathit{Lex} \mid \mathsf{valency}, \mathsf{labels})$ for the set of well-configured Lex-attributed finite trees with $\mathcal{L}$-labeled edges, and $\Phi_{\mathsf{LEX}}(V, \mathcal{L}, \mathit{Lex} \mid \mathsf{valency}, \mathsf{labels})$ for the conjunction of $\Phi_{\mathsf{ID}}(V, \mathcal{L})$ with (1–2); this is again a CSP with the additional functional variable α.

5 Linear Precedence Trees

We assume given a finite set of edge labels $\mathcal{L}_{\text{LP/E}}$, a finite set of node labels $\mathcal{L}_{\text{LP/N}}$, and a total order $\prec$ on their disjoint union $\mathcal{L}_{\text{LP}}$:

$$\mathcal{L}_{\text{LP}} = \mathcal{L}_{\text{LP/E}} \uplus \mathcal{L}_{\text{LP/N}} = \{\ell_1, \ldots, \ell_n\} \qquad \ell_1 \prec \ell_2 \prec \cdots \prec \ell_n$$

A linear precedence tree $G = (V, E, I, <)$ consists of a labeled tree $(V, E) \in \mathbf{T}(V, \mathcal{L}_{\text{LP/E}})$, an assignment $I : V \to \mathcal{L}_{\text{LP/N}}$ of node labels to nodes, and a total order $<$ on V. We say that G is well-ordered if it satisfies the following conditions:

$$w\text{-}\ell_1\!\to_G w_1 \ \wedge\ w\text{-}\ell_2\!\to_G w_2 \ \wedge\ \ell_1 \prec \ell_2 \quad \Rightarrow \quad w_1 < w_2 \tag{3}$$

$$w_1 \to_G^* w_1' \ \wedge\ w_2 \to_G^* w_2' \ \wedge\ w_1 < w_2 \quad \Rightarrow \quad w_1' < w_2' \tag{4}$$

$$w\text{-}\ell_1\!\to_G w_1 \ \wedge\ I(w) = \ell_2 \ \wedge\ \ell_1 \prec \ell_2 \quad \Rightarrow \quad w_1 < w \tag{5}$$

$$w\text{-}\ell_1\!\to_G w_1 \ \wedge\ I(w) = \ell_2 \ \wedge\ \ell_2 \prec \ell_1 \quad \Rightarrow \quad w < w_1 \tag{6}$$

We say that $S \subseteq V$ is $<$-convex in V, and write $\mathrm{convex}(S, V, <)$, iff:

$$\forall w_1, w_2 \in S,\ \forall w_3 \in V \quad w_1 < w_3 < w_2 \ \Rightarrow w_3 \in S$$

We say that G is projective iff:

$$\forall w \in V \quad \mathrm{convex}(\mathsf{eqdown}(w), V, <) \tag{7}$$

We are interested in linear precedence trees which are both projective and well-ordered and write $\mathbf{T}(V, \mathcal{L}_{\text{LP/E}}, \mathcal{L}_{\text{LP/N}}, \prec)$ for those whose node set is V. Again, we are going to characterize the elements of $\mathbf{T}(V, \mathcal{L}_{\text{LP/E}}, \mathcal{L}_{\text{LP/N}}, \prec)$ as the solutions of a CSP.

G induces the additional functions $\mathsf{proj}_G^\ell : V \to 2^V$ for $\ell \in \mathcal{L}_{\text{LP}}$:

$$\mathsf{proj}_G^\ell = \rightarrow_G^* \circ -\ell\rightarrow_G \qquad \text{for } \ell \in \mathcal{L}_{\text{LP/E}}$$

$$\mathsf{proj}_G^\ell(w) = \begin{cases} \{w\} & \text{if} \quad I(w) = \ell \\ \emptyset & \text{otherwise} \end{cases} \qquad \text{for } \ell \in \mathcal{L}_{\text{LP/N}}$$

$\mathsf{proj}_G^\ell(w)$ for $\ell \in \mathcal{L}_{\text{LP/E}}$ is the set of nodes in the subtrees rooted at w's ℓ-daughters. The total order $<$ on V can be extended to a partial order on 2^V as follows:

$$\forall S_1, S_2 \subseteq V,\ S_1 < S_2 \ \equiv\ \forall w_1 \in S_1, \forall w_2 \in S_2\ w_1 < w_2$$

The well-ordering conditions (3–6) are satisfied iff the following property holds:

$$\forall \ell_1, \ell_2 \in \mathcal{L}_{\text{LP}},\ \forall w \in V \qquad \ell_1 \prec \ell_2 \ \Rightarrow\ \mathsf{proj}_G^{\ell_1}(w) < \mathsf{proj}_G^{\ell_2}(w)$$

The projection $\mathsf{proj}_G^\ell(w)$ at an edge label $\ell \in \mathcal{L}_{\text{LP/E}}$ satisfies:

$$\forall w \in V,\ \forall \ell \in \mathcal{L}_{\text{LP/E}} \qquad \mathsf{proj}_G^\ell(w) = \cup\{\mathsf{eqdown}_G(w') \mid w' \in \ell(w)\} \tag{8}$$

i.e. it is the union of the eqdown-sets of w's ℓ-daughters. Each node label $\ell' \in \mathcal{L}_{\text{LP/N}}$ induces a function $\ell' : V \to 2^V$, where $\ell'(w)$ is empty except when ℓ' is the node label $I(w)$ assigned to w, in which case it is $\{w\}$:

$$\{w\} \ =\ \uplus\{\ell'(w) \mid \ell' \in \mathcal{L}_{\text{LP/N}}\} \tag{9}$$

$$w \in \ell'(w) \ \equiv\ I(w) = \ell' \tag{10}$$

Thus the 'projection' $\mathsf{proj}_G^{\ell'}(w)$ at a node label $\ell' \in \mathcal{L}_{\text{LP/N}}$ satisfies:

$$\forall w \in V,\ \forall \ell' \in \mathcal{L}_{\text{LP/N}} \qquad \mathsf{proj}_G^{\ell'}(w) = \ell'(w) \tag{11}$$

The well-ordering conditions are succinctly captured by the following equation:

$$\mathsf{eqdown}(w) \ =\ \mathsf{proj}^{\ell_1}(w) \uplus \cdots \uplus \mathsf{proj}^{\ell_n}(w) \tag{12}$$

which has the declarative semantics of:

$$\mathsf{eqdown}(w) \ =\ \mathsf{proj}^{\ell_1}(w) \uplus \cdots \uplus \mathsf{proj}^{\ell_n}(w)$$
$$\mathsf{proj}^{\ell_1}(w) < \cdots < \mathsf{proj}^{\ell_n}(w)$$

but often allows for stronger inferences. We write $\Phi_{\mathsf{LP}}(V, \mathcal{L}_{\text{LP/E}}, \mathcal{L}_{\text{LP/N}}, \prec)$ for the conjunction of $\Phi_{\mathsf{ID}}(V, \mathcal{L}_{\text{LP/E}})$ with conditions (7–12). This is again a CSP,

with the additional predicate variable $<$ and functional variables I and ℓ' for all $\ell' \in \mathcal{L}_{\text{LP/N}}$. $\mathbf{T}_{\text{LP}}(V, \mathcal{L}_{\text{LP/E}}, \mathcal{L}_{\text{LP/N}}, \prec)$ is in bijection with the solutions of $\Phi_{\mathsf{LP}}(V, \mathcal{L}_{\text{LP/E}}, \mathcal{L}_{\text{LP/N}}, \prec)$.

6 Lexicalized Dependency Grammar

A grammar $\mathcal{G}$ is given by finite sets of labels $\mathcal{L}_{\text{ID}}, \mathcal{L}_{\text{LP/E}}, \mathcal{L}_{\text{LP/N}}$, a total order $\prec$ on $\mathcal{L}_{\text{LP/E}} \uplus \mathcal{L}_{\text{LP/N}}$, a lexicon Lex and functions:

$$\begin{aligned}
\mathsf{valency}_{\text{ID}} &: Lex \to \mathcal{L}_{\text{ID}} \to 2^{\mathbb{N}} \\
\mathsf{valency}_{\text{LP}} &: Lex \to \mathcal{L}_{\text{LP/E}} \to 2^{\mathbb{N}} \\
\mathsf{labels}_{\text{ID}} &: Lex \to 2^{\mathcal{L}_{\text{ID}}} \\
\mathsf{labels}_{\text{LP/E}} &: Lex \to 2^{\mathcal{L}_{\text{LP/E}}} \\
\mathsf{labels}_{\text{LP/N}} &: Lex \to 2^{\mathcal{L}_{\text{LP/N}}} \\
\mathsf{blocks} &: Lex \to 2^{\mathcal{L}_{\text{ID}}}
\end{aligned}$$

$(V, E_{\text{ID}}, E_{\text{LP}}, I, <, \alpha)$ is a valid ID/LP analysis iff it satisfies the conditions below as well as the climbing principles formalized in the next section.

$$\begin{gathered}
(V, E_{\text{ID}}, \alpha) \in \mathbf{T}(V, \mathcal{L}_{\text{ID}}, Lex \mid \mathsf{valency}_{\text{ID}}, \mathsf{labels}_{\text{ID}}) \\
(V, E_{\text{LP}}, \alpha) \in \mathbf{T}(V, \mathcal{L}_{\text{LP/E}}, Lex \mid \mathsf{valency}_{\text{LP}}, \mathsf{labels}_{\text{LP/E}}) \\
(V, E_{\text{LP}}, I, <, \alpha) \in \mathbf{T}(V, \mathcal{L}_{\text{LP/E}}, \mathcal{L}_{\text{LP/N}}, \prec) \\
\forall w \in V \quad I(w) \in \mathsf{labels}_{\text{LP/N}}(\alpha(w))
\end{gathered}$$

7 Climbing Principles

In this section, we formalizes our 'climbing principles' and show that they too can be expressed in terms of set constraints. Given an ID/LP analysis, we use subscripts ID, resp. LP, to distinguish similar variables in the ID tree, resp. the LP tree. For example, we write $\mathsf{mothers}_{\text{ID}}(w)$ for w's mothers in the ID tree and $\mathsf{mothers}_{\text{LP}}(w)$ for its mothers in the LP tree.

Principle 1: A node must land on a transitive head. I.e., $w \to_{\text{LP}} w' \;\Rightarrow\; w \to_{\text{ID}}^{+} w'$. Principle 1 is satisfied whenever:

$$\mathsf{mothers}_{\text{LP}}(w) \subseteq \mathsf{equp}_{\text{ID}}(w) \tag{13}$$

Principle 2: a node may not climb through a barrier. We consider in this article only a very simple notion of barrier: a node may be a barrier to specific syntactic relations; we say that it *blocks* them: e.g. a noun blocks det. The nodes which w must climb through are:

$$\mathsf{through}(w) = \mathsf{up}_{\text{ID}}(w) \;\cap\; \cup\{\mathsf{down}_{\text{ID}}(w') \mid w' \in \mathsf{mothers}_{\text{LP}}(w)\} \tag{14}$$

Principle 2 is satisfied whenever:

$$w''\text{-}\ell\!\rightarrow\! w \in E_{\text{ID}} \;\Rightarrow\; \ell \notin \cup\{\mathsf{blocks}(\alpha(w')) \mid w' \in \mathsf{through}(w)\} \tag{15}$$

Principle 3: a node must land on, or climb higher than its head. This is satisfied whenever:

$$\mathsf{up}_{\text{LP}}(w) \subseteq \cup\{\mathsf{equp}_{\text{LP}}(w') \mid w' \in \mathsf{mothers}_{\text{ID}}(w)\} \tag{16}$$

8 Example Grammar

Our example grammar uses the following sets of labels:

$$\begin{aligned}\mathcal{L}_{\text{ID}} &= \{\mathsf{root}, \mathsf{sbar}, \mathsf{s}, \mathsf{det}, \mathsf{subj}, \mathsf{obj}, \mathsf{vinf}, \mathsf{vpast}, \mathsf{vzu}\}\\ \mathcal{L}_{\text{LP/E}} &= \{\mathsf{df}, \mathsf{vf}, \mathsf{mf}, \mathsf{vc}, \mathsf{xf}\}\\ \mathcal{L}_{\text{LP/N}} &= \{\mathsf{d}, \mathsf{n}, \mathsf{c}, \mathsf{v}\}\end{aligned}$$

df is the determiner field, vf the Vorfeld, mf the Mittelfeld, vc the verbal complement field, and xf the extraposition field. d is an internal field for determiners, n for nouns, c for verb 1st and 2nd (i.e. the complementizer field), v for verb last. The total order $\prec$ is:

$$\mathsf{d} \prec \mathsf{df} \prec \mathsf{n} \prec \mathsf{vf} \prec \mathsf{c} \prec \mathsf{mf} \prec \mathsf{vc} \prec \mathsf{v} \prec \mathsf{xf}$$

Figure 4 contains an example lexicon. For convenience, valencies are described by sets of wildcarded labels. For example, if $\mathsf{valency}(e)$ is described by $\{\ell_1, \ell_2?, \ell_3*\}$, this means:

$$\mathsf{valency}(e)(\ell) = \begin{cases}\{1\} & \text{if } \ell = \ell_1\\ \{0,1\} & \text{if } \ell = \ell_2\\ \{0,1,\ldots,\infty\} & \text{if } \ell = \ell_3\\ \{0\} & \text{otherwise}\end{cases}$$

Mann accepts ID labels {subj, obj} and offers ID valency {det}: it may be subject or object and requires a unique determiner. It accepts LP labels {vf, mf}, i.e. it may land either in the Vorfeld or in the Mittelfeld. It offers a unique field df, for its determiner. Furthermore it blocks ID labels {det}, thus preventing emancipation of its determiner.

For verbs, our grammar distinguishes between coherent (field vc) and extraposed position (field xf). While xf is called the extraposition field, it should not be regarded as uniquely dedicated to the phenomenon of extraposition. For example, both Oberfeldumstellung and extraposition are explained here by migration to the xf field. As we previously described in [4], a number of

phenomena, such as VP extraposition, partial VP extraposition, optional auxiliary flip, V-projection raising, intermediate placement, obligatory auxiliary flip, double auxiliary flip, obligatory coherence can be modeled as emergent from the interaction of our lexicalized constraints.

A finite verb may appear as head of a verb 1st/2nd sentence (internal field c) in which case it typically offers LP valency $\{\mathsf{vf}?, \mathsf{mf}*, \mathsf{vc}?, \mathsf{xf}\}$. It may also appear in a verb last sentence (internal field v) in which case it does not offer vf. Finite verbs block all emancipations.

An infinitive in coherent position offers at most vc. This forces its non-verbal arguments to climb to find a landing place. Only 'zu' infinitives can be extraposed, in which case they typically offer $\{\mathsf{mf}*, \mathsf{vc}?, \mathsf{xf}?\}$ which makes possible full or partial extrapositions.

Ersatzinfinitivs are here modeled as infinitives which can be extraposed.

We have implemented a parser using the formalization described in this article and written a slightly larger grammar than presented here. Our experience so far has been very encouraging and confirms the practical effectiveness of constraint propagation. For example, parsing the sentence "daß Maria einen Mann wird lieben können" requires no search. Our parser can also function in a mode where it disregards the linear order of its input and instead generates all possible linearizations. Figure 5 shows an example of this applied to the same sentence: the window on the left contains the search tree; the 7 lineralizations are enumerated optimally (without failures). The window on the right displays one analysis: the ID tree is shown above and the LP tree below.

9 Conclusion

We described a lexicalized formulation of dependency grammar where an analysis consists of two mutually constraining trees: a non-ordered non-projective tree of syntactic dependencies (ID tree) and an ordered projective tree of topological dependencies (LP tree). Both trees are subject to similar lexicalized configuration constraints. Additionally the shape of the LP tree is a flattening of the ID tree's obtained by allowing nodes to climb up subject to 3 principles.

We precisely formalized the well-formedness conditions characterizing valid analyses. Furthermore, this formalization can also be regarded as a constraint program and forms the basis of our implementation in Oz.

References

[1] Bech, G., "Studien über das deutsche Verbum infinitum," Munksgaard, Kopenhagen, 1955.

[2] Bröker, N., "Eine Dependenzgrammatik zur Kopplung heterogener Wissensquellen," Linguistische Arbeiten 405, Max Niemeyer Verlag, Tübingen, 1999.

word	syntax		topology			
	$\text{labels}_{\text{ID}}$	$\text{valency}_{\text{ID}}$	$\text{labels}_{\text{LP/E}}$	$\text{labels}_{\text{LP/N}}$	$\text{valency}_{\text{LP}}$	blocks
daß	{sbar}	{s}	{vf, xf}	{c}	{vc}	{}
einen	{det}	{}	{df}	{d}	{}	{}
Mann	{subj, obj}	{det}	{vf, mf}	{n}	{df}	{det}
Maria	{subj, obj}	{}	{vf, mf}	{n}	{}	{}
lieben	{vinf}	{obj}	{vc}	{v}	{}	{}
zu lieben	{vzu}	{obj}	{vc}	{v}	{}	{}
zu lieben	{vzu}	{obj}	{xf}	{v}	{mf∗, xf?}	{}
können	{vinf}	{vinf}	{vc}	{v}	{vc?}	{}
können	{vinf, vpast}	{vinf}	{xf}	{v}	{mf∗, vc?, xf?}	{}
hat	{root}	{subj, vpast}	{vc}	{c}	{vf?, mf∗, vc?, xf?}	$\mathcal{L}_{\text{ID}}$
hat	{s}	{subj, vpast}	{vc}	{v}	{mf∗, vc?, xf?}	$\mathcal{L}_{\text{ID}}$
wird	{root}	{subj, vinf}	{vc}	{c}	{vf?, mf∗, vc?, xf?}	$\mathcal{L}_{\text{ID}}$
wird	{s}	{subj, vinf}	{vc}	{v}	{mf∗, vc?, xf?}	$\mathcal{L}_{\text{ID}}$
haben	{vinf}	{vpast}	{xf}	{v}	{mf∗, vc?, xf?}	{}
versucht	{root}	{subj, vzu}	{vc}	{c}	{vf?, mf∗, vc?, xf?}	$\mathcal{L}_{\text{ID}}$
versucht	{sbar}	{subj, vzu}	{vc}	{v}	{mf∗, vc?, xf?}	$\mathcal{L}_{\text{ID}}$

Fig. 4. Example Lexicon

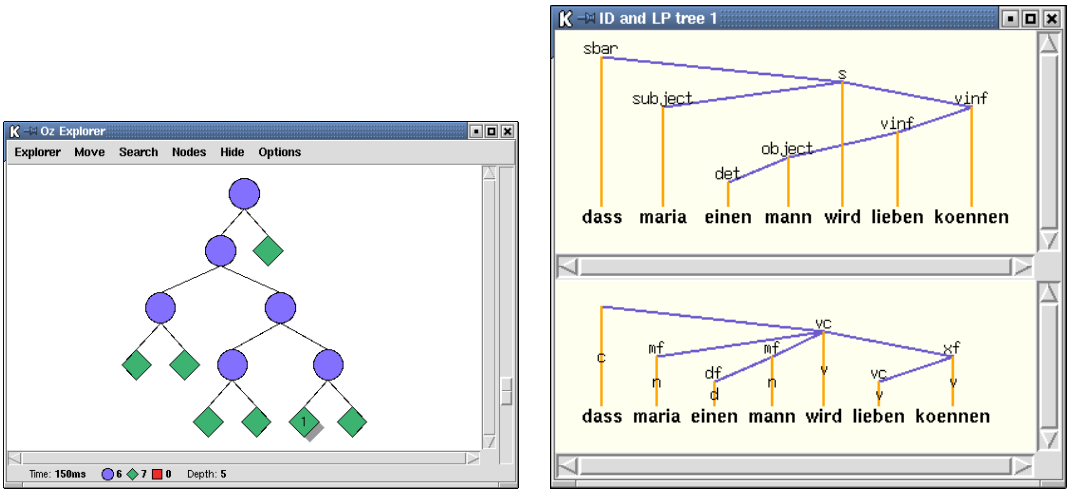

Fig. 5. Searching for all linearizations of *"daß Maria einen Mann wird lieben können"* and displaying one specific analysis

[3] Duchier, D., *Axiomatizing dependency parsing using set constraints*, in: *Sixth Meeting on Mathematics of Language*, Orlando, Florida, 1999, pp. 115–126.

[4] Duchier, D. and R. Debusmann, *Topological dependency trees: A constraint-based account of linear precedence*, in: *39th Annual Meeting of the Association for Computational Linguistics (ACL 2001)*, Toulouse, France, 2001.

[5] Kahane, S., A. Nasr and O. Rambow, *Pseudo-projectivity: a polynomially*

parsable non-projective dependency grammar, in: *Proceedings of ACL/COLING '98*, Montréal, 1998, pp. 646–52.

[6] Kathol, A., "Linearization-Based German Syntax," Ph.D. thesis, Ohio State University (1995).

[7] Müller, S., "Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche," Linguistische Arbeiten 394, Max Niemeyer Verlag, Tübingen, 1999.

[8] Reape, M., *Domain union and word order variation in German*, in: J. Nerbonne, K. Netter and C. Pollard, editors, *German in Head-Driven Phrase Structure Grammar*, CSLI, Stanford/CA, 1994 pp. 151–197.

[9] Uszkoreit, H., "Word Order and Constituent Structure in German," CSLI, Stanford/CA, 1987.