11

A Hierarchy of Mildly Context-Sensitive Dependency Grammars

ANSSI YLI-JYRÄ AND MATTI NYKÄNEN

Dept. of General Linguistics and Dept. of Computer Science,
University of Helsinki, Finland
Email: Anssi.Yli-Jyra@ling.helsinki.fi, Matti.Nykanen@cs.helsinki.fi

ABSTRACT. The paper presents Colored Multiplanar Link Grammars (CMLG). These grammars are reducible to extended right-linear S-grammars (Wartena 2001) where the storage type S is a concatenation of c pushdowns. The number of colors available in these grammars induces a hierarchy of Classes of CMLGs. By fixing also another parameter in CMLGs, namely the bound t for *non-projectivity depth*, we get c-Colored t-Non-projective Dependency Grammars (CNDG) that generate acyclic dependency graphs. Thus, CNDGs form a two-dimensional hierarchy of dependency grammars. A part of this hierarchy is mildly context-sensitive and non-projective.

11.1 Introduction

This paper proposes non-projective, polynomially parseable lexicalized grammars capable of describing scrambling and long-distance dependencies up to a bounded *nested crossing depth* and a bounded *non-projectivity depth.*

In terms of dependency grammar (DG) (Tesnière 1959), word-order is distinct from the dependency tree that analyzes the structure of the sentence. However, Hays (1964) and Gaifman (1965) have formalized Tesnière's ideas so that their DGs describe only *projective* linearisations.

What we would like to have is a mildly context-sensitive (MCS) (Joshi 1985) superclass of the Hays-Gaifman DGs that captures also

[Pre-]*Proceedings of Formal Grammar 2004.* [PDF-LaTeX rerun by A. Yli-Jyrä, 2005]
Gerhard Jäger, Paola Monachesi, Gerald Penn, and Shuly Wintner (eds.).
Copyright © 2004, the individual authors.

non-projective dependencies, e.g. scrambling, the possibility of the elements of a sentence to lie in arbitrary permutations. However, it seems that grammars that capture unrestricted scrambling fail to be mildly context-sensitive, *cf.* Global Index Grammar (GIG) (Castaño 2003). Limited scrambling is, however, captured by linear context-free rewriting systems (LCFRSs) (Vijay-Shanker et al. 1987) that are currently the best characterisation for MCS grammars.

Linear Indexed Grammar (LIG) (Gazdar 1988) is a LCFRS that represents nested non-local dependencies through an *index pushdown* that is associated with nodes in derivation trees. The additional power of some other LCFRSs is based on replacing the index pushdowns with *index storages* of a more general type $\mathcal{S}$. We will base our investigations on extended right-linear $\mathcal{S}^c_{\text{pd}}$-grammars (ERL-$\mathcal{S}$-Gs) Wartena (2001) whose storage type consists of c pushdowns.

In this paper, some new restrictions on ERL-$\mathcal{S}^k_{\text{pd}}$-Gs are developed. Through these restrictions we obtain various classes of DGs. The obtained DGs can be used to describe restricted non-projective dependencies and restricted scrambling, and they contain the Hays-Gaifman DGs as a subclass.

The important contribution of this paper is to show that when we set bounds for *nested crossing depth* and *non-projectivity depth*, we obtain classes of DGs that are mildly context-sensitive. The length of a longest chain ( ) of crossing dependencies constitute a lower bound for the *nested crossing depth* that is defined, in this paper, as the number of concatenated pushdowns c needed in derivation. By the *non-projectivity depth*, we mean the number of times dependency links climb from a projective position to a non-projective one along a path of directed dependencies.

The paper is structured as follows. Section 11.2 defines Context-Free Linear $\mathcal{S}^{c,\Gamma}_{\text{pd}}$-Grammars with Extended Domain of Locality. Section 11.3 eliminates ambiguity that is related to the storage allocation. In Section 11.4, we introduce *Colored Multiplanar Link Grammars* (CMLG), and discuss their properties in Section 11.5. In Section 11.6 we enforce a sufficient condition for acyclicity in c-Colored t-Non-projective Dependency Grammars (CNDG) that form a sub-hierarchy among CMLGs. The conclusion is in Section 11.8.

11.2 The Basic Machinery

11.2.1 Storage Type

Definition 1 A *storage type* is a tuple $\mathcal{S} = (C, C_{\text{i}}, C_{\text{f}}, \Phi, \Pi, m)$, where

- C is the set of *configurations*, and $C_{\text{i}}, C_{\text{f}} \subseteq C$ are respectively the sets of *initial* and *final* configurations,

- Φ and Π are respectively the sets of *instructions* and *predicates*,
- m is the *meaning function.* It associates to each $\pi \in \Pi$ the corresponding function $m(\pi)\colon C \to \{\text{TRUE, FALSE}\}$, and to each $\phi \in \Phi$ the corresponding partial function $m(\phi)\colon C \to C$.

The meaning function m is extended to Boolean combinations of the predicates Π in the natural way and to nonempty strings $\phi = (\Phi \cup (\mathcal{B}^\Pi \times \Phi^+))^+$ and pairs $(\pi, \phi) \in (\mathcal{B}^\Pi \times \Phi^+)$ by defining $m(\phi_1\phi_2)(\kappa) = m(\phi_2)(m(\phi_1)(\kappa))$, where $\kappa \in C$ and by defining

$$m((\pi,\phi))(\kappa) = \begin{cases} m(\phi)(\kappa), & \text{if } m(\pi)(\kappa) = \text{TRUE},\\ \kappa, & \text{otherwise.}\end{cases}$$

Wartena (2001) defines a trivial (memoryless) storage $\mathcal{S}_{\text{triv}}$, an ordinary pushdown $\mathcal{S}_{\text{pd}}$ and concatenations on storage types. The concatenation w.r.t. writing is denoted as $\circ_w$.

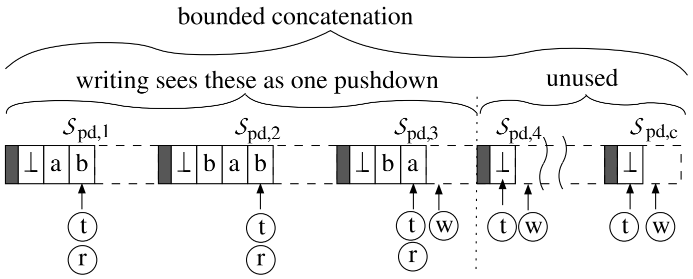
bounded concatenation
writing sees these as one pushdown
unused
$\mathcal{S}_{\text{pd},1}$
$\mathcal{S}_{\text{pd},2}$
$\mathcal{S}_{\text{pd},3}$
$\mathcal{S}_{\text{pd},4}$
$\mathcal{S}_{\text{pd},c}$
⊥ a b
⊥ b a b
⊥ b a
⊥
⊥
t
r
w

FIGURE 1 Example of a possible configuration of a storage $(...(((\mathcal{S}_{\text{triv}} \circ_w \mathcal{S}_{\text{pd},1}) \circ_w \mathcal{S}_{\text{pd},2}) \circ_w \mathcal{S}_{\text{pd},3}) \ldots) \circ_w \mathcal{S}_{\text{pd},c}$ with marking of possibilities of applying operations TOP (t), POP (r) and PUSH (w).

11.2.2 Concatenating Storage Type

We restrict our attention to storages of type $(...(((\mathcal{S}_{\text{triv}} \circ_w \mathcal{S}_{\text{pd},1}) \circ_w \mathcal{S}_{\text{pd},2}) \circ_w \mathcal{S}_{\text{pd},3}) \ldots) \circ_w \mathcal{S}_{\text{pd},c}$ that is intuitively a tuple $\langle \mathcal{S}_{\text{pd},1}, \mathcal{S}_{\text{pd},2}, \mathcal{S}_{\text{pd},3}, \ldots, \mathcal{S}_{\text{pd},c}\rangle$ of c pushdowns $\mathcal{S}_{\text{pd},p}$ (Figure 1) with the restriction that writing new elements into pushdown $\mathcal{S}_{\text{pd},p}$ is permitted only if all the succeeding pushdowns $\mathcal{S}_{\text{pd},p+1}$, $\mathcal{S}_{\text{pd},p+2}$, $\mathcal{S}_{\text{pd},p+3}$, $\ldots$, $\mathcal{S}_{\text{pd},c}$ are empty. Otherwise each pushdown $\mathcal{S}_{\text{pd},p}$ can be used independently of the others. More formally, this storage type is defined as follows:

Definition 2 A *writing-concatenating tuple of c pushdowns over a stack alphabet* Γ is the following storage type $\mathcal{S}_{\text{pd}}^{c,\Gamma} = (C, C_{\text{i}}, C_{\text{f}}, \Phi, \Pi, m)$:

- The configurations are $C = ((\Gamma \cup \{\sharp, \flat, \natural\})^* \bot)^c$, where $\bot \notin \Gamma$ is a special symbol denoting the bottom of the pushdown, and $\sharp, \flat, \natural \notin \Gamma$ are special *semaphore* symbols reserved for the restriction of normalized $\mathcal{S}_{\mathrm{pd}}^{c,\Gamma}$ that is introduced in Section 11.3,
- the unique initial and final configuration is $C_{\mathrm{i}} = C_{\mathrm{f}} = \{\bot\}^c$,
- the predicates are $\Pi = \{\textsc{top}_p(a) \mid 1 \leq p \leq c, a \in \Gamma \cup \{\bot, \sharp, \flat, \natural\}\}$,
- the instructions are $\Phi = \{\textsc{id}, \textsc{undef}\} \cup \{\textsc{push}_p(\beta) \mid 1 \leq p \leq c, \beta \in (\Gamma \cup \{\sharp, \flat, \natural\})^+\} \cup \{\textsc{pop}_p(\beta) \mid 1 \leq p \leq c, \beta \in (\Gamma \cup \{\sharp, \flat, \natural\})^+\}$.

The predicates Π and instructions Φ have the following basic meanings:

$$m(\textsc{id})(\langle \alpha_1, \cdots, \alpha_c \rangle) = \langle \alpha_1, \cdots, \alpha_c \rangle$$
$$m(\textsc{top}_p(a))(\langle \alpha_1, \cdots, \alpha_{p-1}, \beta b, \alpha_{p+1}, \cdots, \alpha_c \rangle) = (a = b)$$
$$m(\textsc{push}_p(\beta))(\langle \alpha_1, \cdots, \alpha_p, \bot, \cdots, \bot \rangle) = \langle \alpha_1, \cdots, \alpha_{p-1}, \alpha_p \beta, \bot, \cdots, \bot \rangle,$$
$$m(\textsc{pop}_p(\beta))(\langle \alpha_1, \cdots, \alpha_{p-1}, \alpha_p \beta^{\mathrm{r}}, \alpha_{p+1}, \cdots, \alpha_c \rangle) = \langle \alpha_1, \cdots, \alpha_c \rangle$$

where β^r is the reverse of any string $\beta \in (\Gamma \cup \{\sharp, \flat, \natural\})^+$ and the functions $m(\phi) : C \to C$, $\phi \in \Phi$, remain undefined for all other cases.

11.2.3 Context-free Linear-$\mathcal{S}$-Grammars

Definition 3 Let $\mathcal{S} = (C, C_{\mathrm{i}}, C_{\mathrm{f}}, \Phi, \Pi, m)$ be a storage type. A *context-free linear $\mathcal{S}$-grammar with extended domain of locality (CFL-EDL-$\mathcal{S}$-G)* is a tuple $G = (V_N, V_T, P, S, \kappa_0)$, where

- the pairwise disjoint finite sets V_N and V_T are the *nonterminal* and *terminal alphabets*, respectively,
- $S \in V_N$ is the *start symbol*, and $\kappa_0 \in C_{\mathrm{i}}$ is the *start configuration*, and
- P is a finite set of *productions* of the form

$$X\phi_1 \to \text{if } \pi \text{ then } \zeta_1 \, Y\phi_2 \, \zeta_2 \tag{11.1}$$
$$X \to \text{if } \pi \text{ then } w \tag{11.2}$$

  where $X, Y \in V_N$, $\phi_1 \in (\mathcal{B}^{\Pi} \times \Phi^+)^+$, $\phi_2 \in \Phi^+$, $\pi \in \mathcal{B}^{\Pi}$, $\zeta_1, \zeta_2 \in (V_N \cup V_T)^*$, and $w \in V_T^*$.

The set $\mathbf{S} = ((V_N \times C) \cup V_T)^*$ is called the set of *sentential forms*, and $\sigma \in \mathbf{S}$ is said to *derive* $\tau \in \mathbf{S}$ if and only if $\sigma = \alpha \, (X, \kappa) \, \beta$ and $\tau = \alpha\gamma\beta$ for some $\alpha, \beta, \gamma \in \mathbf{S}$ and P contains either

- a production of the type (11.1) for which $m(\phi_1(\neg\pi, \textsc{undef})\phi_2)(\kappa)$ is defined and $\gamma = \zeta_1' \; (Y, m(\phi_1\phi_2) \; (\kappa)) \; \zeta_2'$, where ζ_1' and ζ_2' are obtained from ζ_1 and ζ_2 respectively by replacing every nonterminal D by (D, κ_0), or,
- a production of the type (11.2) for which $m(\pi)(\kappa) = \textsc{true}$ and $\gamma = w$.

The initial sentential form is $\langle (S, \kappa_0) \rangle$. The *derivations* and the *generated language* of grammar G are defined in a usual way.

Definition 4 A CFL-EDL-$\mathcal{S}$-G is a *right-linear $\mathcal{S}$-grammar* with extended domain of locality (RL-EDL-$\mathcal{S}$-G), if its productions of the form (11.1) are such that $\zeta_1 \in V_T^*$ and $\zeta_2 \in \epsilon$.

A *context-free linear $\mathcal{S}$-grammar* (CFL-$\mathcal{S}$-G) (Weir 1994) is a CFL-EDL-$\mathcal{S}$-grammar, whose productions of the form (11.1) are such that $\phi_1 = \epsilon$ and $\phi_2 \in \Phi$.

A *right linear $\mathcal{S}$-grammar* (RL-$\mathcal{S}$-G) is an RL-EDL-$\mathcal{S}$-G, whose productions of the form (11.1) are such that $\phi_1 = \epsilon$ and $\phi_2 \in \Phi$.

An *extended right-linear $\mathcal{S}$-grammar* (ERL-$\mathcal{S}$-G) (Wartena 2001) is a CFL-$\mathcal{S}$-G, whose productions of the form (11.1) are such that $\zeta_1 \in V_N$ and $\zeta_2 \in V_T \cup \{\epsilon\}$.

Theorem 1 *CFL-EDL-$\mathcal{S}$-Gs and CFL-$\mathcal{S}$-Gs generate the same languages, and RL-EDL-$\mathcal{S}$-Gs and RL-$\mathcal{S}$-Gs generate the same languages.*

Proof. The inclusions $\mathcal{L}$(CFL-$\mathcal{S}$-G) $\subseteq$ $\mathcal{L}$(CFL-EDL-$\mathcal{S}$-G) and $\mathcal{L}(RL - EDL - \mathcal{S} - G) \subseteq \mathcal{L}$(RL-$\mathcal{S}$-G) follows from the definition of the grammars. To show that the reverse inclusions hold, we replace productions of the form (11.1) by expanding them syntactically into

$$X \to \text{if} \quad \textsc{true} \quad \text{then} \quad \zeta_1\, Q^Y_{\phi_1 \pi \phi_2}\, \zeta_2$$

and define the productions for new nonterminals Q^Y_ω inductively as

$$\begin{aligned}
Q^Y_\epsilon &\to \text{if} \quad \textsc{true} \quad \text{then} \quad Y \ \textsc{id} \\
Q^Y_{\langle \pi', \phi' \rangle \omega} &\to \text{if}\, \pi' \,\text{then}\ Q^Y_{\phi' \omega}\ \textsc{id} \\
Q^Y_{\langle \pi', \phi' \rangle \omega} &\to \text{if} \neg\, \pi' \,\text{then}\ Q^Y_{\omega}\ \ \textsc{id} \\
Q^Y_{\pi' \omega} &\to \text{if}\, \pi' \,\text{then}\, Q^Y_{\omega}\ \ \textsc{id} \\
Q^Y_{\phi' \omega} &\to \text{if} \quad \textsc{true} \quad \text{then} \quad Q^Y_{\omega} \phi'
\end{aligned}$$

where $\pi' \in \mathcal{B}^{\Pi}$, $\phi' \in \Phi^+$, and where ω denotes suffixes of the three-part string $\phi_1 \pi \phi_2$. All the productions of the form (11.1) in the expanded grammar contain only productions where $\phi_1 = \epsilon$ and $\phi_2 \in \Phi$. The expanded grammar recognizes the language of the original grammar. □

Note that a derivation step of the original CFL-EDL-$\mathcal{S}$-G may corresponds to multiple steps in the resulting CFL-$\mathcal{S}$-G.

Theorem 2 *Every RL-$\mathcal{S}$-G can be reduced to an ERL-$\mathcal{S}$-G.*

Proof. A new nonterminal and a new production are created for each non-empty prefix of $\zeta_1 \in V_T^+$. This allows replacing $\zeta_1 \in V_T^+$ in the original productions with $\zeta'_1 \in V'_N$ in the productions of the ERL-$\mathcal{S}$-G. □

The following two properties of ERL-$\mathcal{S}$-Gs are needed in Section 11.5:

Proposition 3 *ERL-$\mathcal{S}$-G is a linear context-free rewriting system. Thus, it is polynomially parseable and has linear growth property.*

11.3 Normalized Storage

If the number of pushdowns is larger than the number of nested crossing dependencies in derivations, CFL-EDL-$\mathcal{S}^{c,\Gamma}_{\mathrm{pd}}$-Gs have some freedom in allocation of different pushdowns for different stack symbols. An example of this is shown in Figure 2. Some ways to allocate pushdowns

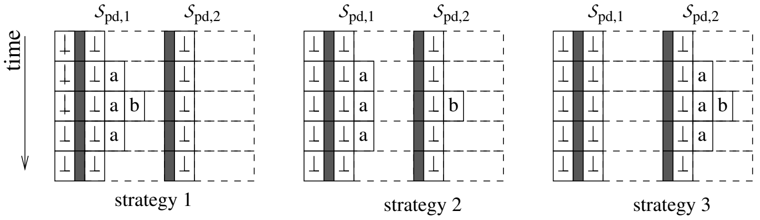

FIGURE 2 An example of ambiguity in storage allocation.

have already been banned by the the following restrictions that are imposed by the concatenating storage type $\mathcal{S}^{c,\Gamma}_{\mathrm{pd}}$:

1. the LIFO discipline applies separately to each pushdown, and
2. the pushdowns are concatenated with respect to writing.

In addition to the effect of these two restrictions, we want to eliminate strategies 2 and 3 shown in Figure 2. For this purpose, we propose the following restrictions that eliminate this kind of ambiguity in allocation of pushdowns:

3. An operation that writes to an empty pushdown $\mathcal{S}_{\mathrm{pd},p}$, $p \geq 2$, is allowed only if $\mathcal{S}_{\mathrm{pd},p-1}$ is nonempty.
4. An operation that writes to an empty pushdown $\mathcal{S}_{\mathrm{pd},p}$, $p \geq 2$, is allowed only if the current configuration contains a stack symbol in $\mathcal{S}_{\mathrm{pd},p-1}$ that will be read before $\mathcal{S}_{\mathrm{pd},p}$ becomes empty again.

Let Φ' be an extended set of instructions on the normalized storage type Norm$\mathcal{S}^{c,\Gamma}_{\mathrm{pd}}$. It is the union of Φ and $\{\textsc{rpush}_p(\beta) \mid 1 \leq p \leq c, \beta \in \Gamma^+\} \cup \{\textsc{rpop}_p(a) \mid 1 \leq p \leq c, a \in \Gamma\}$, where the new RPUSH and RPOP instructions obey the constraints 3 and 4, while the old PUSH and POP instructions do not.

The meanings of the new instructions are defined by means of semaphore symbols: when an RPUSH$_p$ instruction to an empty pushdown p, $p > 1$, takes place, a semaphore symbol $\flat$ is written on the bottom of $\mathcal{S}_{\mathrm{pd},p}$ and two semaphore symbols $\sharp$ and $\natural$ are written respectively on the top of pushdowns $\mathcal{S}_{\mathrm{pd},p-1}$ and $\mathcal{S}_{\mathrm{pd},p}$. The symbol $\natural$ is kept always on the top of $\mathcal{S}_{\mathrm{pd},p}$. Later, when a normal symbol is being read from the pushdown $\mathcal{S}_{\mathrm{pd},p-1}$ with RPOP, these semaphores $\sharp$ and $\natural$ are first removed respectively from $\mathcal{S}_{\mathrm{pd},p-1}$ and $\mathcal{S}_{\mathrm{pd},p}$ if they have not yet been removed. When the semaphore symbol $\flat$ is read from $\mathcal{S}_{\mathrm{pd},p}$, this must happen immediately after reading a non-semaphore symbol (rather than $\natural$) $\mathcal{S}_{\mathrm{pd},p}$. If the semaphores cannot be read in this way, there are no other ways to read them. Thus, the derivation will be stuck in the cases where the restrictions 3 and 4 cannot be satisfied. More formally, the meaning functions are:

$$m(\textsc{rpush}_p(\beta))(\kappa) =$$

$$\begin{cases} m(\textsc{push}_{p\text{-}1}(\sharp)\ \textsc{push}_p(\flat\beta\natural))(\kappa)\}, & \text{if } p \geq 2 \quad \vee\ \textsc{top}_p(\perp)(\kappa); \\ m(\textsc{pop}_p(\natural)\ \textsc{push}_p(\beta\natural))(\kappa), & \text{if } p \geq 2 \quad \vee\ \textsc{top}_p(\natural)(\kappa); \\ m(\textsc{push}_p(\beta))(\kappa), & \text{otherwise,} \end{cases}$$

and $m(\textsc{rpop}_p(a))(\kappa) =$

$$\begin{cases} m(\textsc{pop}_p(a))(\kappa), & \text{if } p = 1; \\ m(\ (\textsc{top}_p(\sharp) \wedge \textsc{top}_{p+1}(\natural), \textsc{pop}_p(\sharp)\ \textsc{pop}_{p+1}(\natural)) & \\ \quad (\textsc{top}_p(\natural),\ \textsc{pop}_p(\natural a)\ \textsc{push}_p(\natural)\,)\ (\neg\,\textsc{top}_p(\natural),\ \textsc{pop}_p(a)) & \\ \quad (\textsc{top}_p(\flat),\ \textsc{pop}_p\,(\flat))\ )\ (\kappa), & \text{otherwise.} \end{cases}$$

Theorem 4 *CFL-EDL-$\mathcal{S}_{\mathrm{pd}}^{c,\Gamma}$-Gs using* RPUSH *and* RPOP *instructions can be reduced to CFL-EDL-$\mathcal{S}_{\mathrm{pd}}^{c,\Gamma}$-Gs that don't use these instructions.*

Proof. These new instructions can be regarded as shorthand notations which extend the transformation in Theorem 1 as follows:

Nonterminals $Q^Y_{\textsc{rpush}_1(\beta)\omega}$, $Q^Y_{\textsc{rpop}_1(\beta)\omega}$ and $Q^Y_{\textsc{rpop}_p(\beta)\omega}$ where $p \geq 2$ are replaced respectively with nonterminals $Q^Y_{\textsc{push}_1(\beta)\omega}$, $Q^Y_{\textsc{pop}_1(\beta)\omega}$ and $Q_{(\textsc{top}_p(\sharp) \wedge \textsc{top}_{p+1}(\natural), \textsc{pop}_p(\sharp)\ \textsc{pop}_{p+1}(\natural))\quad (\textsc{top}_p(\natural),\ \textsc{pop}_p(\natural a)\ \textsc{push}_p(\natural))\ (\neg\,\textsc{top}_p(\natural),\ \textsc{pop}_p(a))\ (\textsc{top}_p(\flat),\ \textsc{pop}_p(\flat))}$. Nonterminals $Q^Y_{\textsc{rpush}_p(\beta)\omega}$, where $p \geq 2$, create the grammar rules

$$Q^Y_{\textsc{rpush}_p(\beta)\omega} \rightarrow \text{if } \textsc{top}_p(\perp) \text{ then } Q^Y_\omega\ \textsc{push}_{p-1}(\sharp)\ \textsc{push}_p(\flat\beta\natural)$$

$$Q^Y_{\textsc{rpush}_p(\beta)\omega} \rightarrow \text{if } \textsc{top}_p(\natural) \text{ then } Q^Y_\omega\ \textsc{pop}_p(\natural)\ \textsc{push}_p(\beta\natural)$$

$$Q^Y_{\textsc{rpush}_p(\beta)\omega} \rightarrow \text{if } \neg\,\textsc{top}_p(\perp) \wedge \neg\,\textsc{top}_p(\natural) \text{ then } Q^Y_\omega\ \textsc{push}_p(\beta). \qquad \square$$

Definition 5 A *Context-free linear normalized $\mathcal{S}^{c,\Gamma}_{\mathrm{pd}}$ grammar with extended domain of locality* (CFL-EDL-Norm$\mathcal{S}^{c,\Gamma}_{\mathrm{pd}}$-G) is a CFL-EDL-$\mathcal{S}^{c,\Gamma}_{\mathrm{pd}}$- grammar whose rules do not directly use PUSH and POP instructions, but use RPUSH and RPOP instead.

Theorem 5 *CFL-EDL-Norm$\mathcal{S}^{c,\Gamma}_{\mathrm{pd}}$-Gs allocate pushdowns for different stack symbols so that they conform the restrictions 1 - 4.*

Proof. The proof is omitted for brevity. □

Dependencies are binary relations between string positions of the generated string. They are described by symbols that are written to a pushdown at one string position and read from that pushdown at another position. The reason for having multiple pushdowns in the storage Norm$\mathcal{S}^{c,\Gamma}_{\mathrm{pd}}$ is to enable capturing *nested crossing dependencies.* For any set of dependency links whose starting and finishing times are disjoint, there is only one way to allocate Norm$\mathcal{S}^{c,\Gamma}_{\mathrm{pd}}$ for these links. Yli-Jyrä (2003) has experimented with a data structure equivalent to Norm$\mathcal{S}^{c,\Gamma}_{\mathrm{pd}}$ and shown that dependency trees of a small dependency treebank can be represented as a sequence of configurations of such a data structure. This motivates introduction of the grammar formalisms of Sections 11.4 and 11.6.

11.4 Colored Multiplanar Link Grammar (CMLG)

Definition 6 A *nonterminal-free Norm$\mathcal{S}^{c,\Gamma}_{\mathrm{pd}}$-grammar* is a *RL−EDL−Norm$\mathcal{S}^{c,\Gamma}_{\mathrm{pd}}$ − G* $G = (V_N, V_T, P, S, \kappa_0)$ for which $V_N = \{S, X\}$ and whose productions are of the following forms:

$$\begin{array}{llll} S & \to \text{if} \quad \mathrm{TRUE} & \text{then} \quad X\phi_{2,1} & (11.3)\\ X\phi_{1,r} & \to \text{if} \quad \wedge_{1\le p\le c}\,\mathrm{TOP}_p(\bot) & \text{then} \quad \epsilon & (11.4)\\ X\phi_{1,r} & \to \text{if} \quad \pi & \text{then}\, aX\phi_{2,1} & (11.5)\\ X\phi_{1,r} & \to \text{if} \quad \pi & \text{then} \quad X\phi_{2,r+1} & (11.6) \end{array}$$

where $a \in V_T$, $\pi \in \mathcal{B}^{\Pi}$, $1 \le r \le c$, $\phi_{1,r} \in \{\mathrm{RPOP}_p(a) \mid 1 \le p \le r, a \in \Gamma\}^*$, and $\phi_{2,s} \in \{\mathrm{RPUSH}_q(\alpha) \mid s \le q \le c, \alpha \in \Gamma^*\}$.

Definition 7 A *colored multiplanar link grammar (CMLG)* is a structure $G = \langle V_T, \Lambda_D, \Lambda_H, c, \Psi\rangle$ where V_T is the set of *terminal symbols*, Λ_D and $\Lambda_H = \{\overline{a} \mid a \in \Lambda_D\}$ are respectively the sets of *dependent* and *governor labels*, c is the *number of colors*, and Ψ is the set of *colored rules* of the forms

$$*(Y_1 \ \ldots \ Y_m) \tag{11.7}$$

$$(p_1/V_1 \ \ldots \ p_n/V_n)* \tag{11.8}$$

$$a(p_1/V_1 \ \ldots p_n/V_n \ * \ q/Y_1 \ Y_2 \ \ldots \ Y_m) \tag{11.9}$$

$$0(p_1/V_1 \ \ldots p_n/V_n \ * \ q/Y_1 \ Y_2 \ \ldots \ Y_m) \tag{11.10}$$

where $a \in V_T$, $0 \notin V_T$, and $V_1, V_2, \ldots, V_n, Y_1, Y_2, \ldots Y_m \in \Lambda_D \cup \Lambda_H$, $p_1, p_2, \ldots, p_n, q \in [1, 2, .., c]$ and $p_i \leq p_{i+1}$ for $1 \leq i \leq n$. Moreover, in rules of type (11.10)[1] it holds that $\max\{p_1, p_2, \ldots, p_n\} + 1 \leq q \leq c$.

The semantics of the grammar G is defined by reducing it to a nonterminal-free Norm$\mathcal{S}^{c,\Gamma}_{\mathrm{pd}}$-grammar $G = (V_N,\ V_T,\ P,\ S,\ \kappa_0)$, with stack alphabet $\Gamma = \{(\overleftarrow{x}, \_), (\overrightarrow{x}, \_) \mid x \in \Lambda_D\}$ and set P containing the following productions:

$$S \to \text{if } \textsc{true} \text{ then } X \ \textsc{push}_1(y_m y_{m-1} \cdots y_1)$$

for each rule of type (11.7);

$$X \ \textsc{pop}_{p_n}(v_n) \cdots \textsc{pop}_{p_1}(v_1) \to \text{if } \wedge_{1 \leq p \leq c} \textsc{top}_p(\bot) \text{ then } \epsilon$$

for each rule of type (11.8); and respectively

$$X \ \textsc{pop}_{p_n}(v_n) \cdots \textsc{pop}_{p_1}(v_1) \to \text{if } \textsc{true} \text{ then } aX \ \textsc{push}_q(y_m y_{m-1} \cdots y_1),$$
$$X \ \textsc{pop}_{p_n}(v_n) \cdots \textsc{pop}_{p_1}(v_1) \to \text{if } \textsc{true} \text{ then } X \ \textsc{push}_q(y_m y_{m-1} \cdots y_1)$$

for each rule of type (11.9) and (11.10), where $v_i \in \lambda(V_i)$ and $y_i \in \rho(Y_i)$ and functions $\lambda, \rho : (\Lambda_D \cup \Lambda_H) \to 2^\Gamma$ are defined as follows:

$$\lambda(\overline{x}) = \{(\overrightarrow{x}, \_)\} \qquad \rho(\overline{x}) = \{(\overleftarrow{x}, \_)\}$$
$$\lambda(x) = \{(\overleftarrow{x}, \_)\} \qquad \rho(x) = \{(\overrightarrow{x}, \_)\},$$

11.5 Mild Context-Sensitivity of CMLGs

The notion of mild context-sensitivity is an attempt by Joshi (1985) to express the formal power needed to define natural languages.

Definition 8 A class of grammars is *mildly context-sensitive* if the grammars of this class are (i) polynomial time parseable and (ii) they capture multiple dependencies, limited crossing dependencies and the copy language and its grammars generate (iii) a proper superclass of context-free languages where (iv) all the languages have linear growth property.

Theorem 6 *$\mathbf{G}_c$, where $c \geq 2$, is mildly context-sensitive.*

Proof. We will now prove that $\mathbf{G}_c$ has the properties (i) - (iv).

[1]If we drop the rules of the form (11.10), we will not be able to express the copy language as required for MCS grammars.

(i) As shown previously, each CMLG $G \in \mathbf{G}_c$ can be reduced to a polynomial size ERL-$\mathcal{S}^{c,\Gamma}_{\mathrm{pd}}$-G, and the latter is known to be polynomially parseable. The class $\mathbf{G}_c$ are, thus, polynomially parseable.

(ii) The capability to describe multiple dependencies is usually represented as an ability to describe languages $L_1 = \{a^n b^n c^n \mid n \geq k\}$, $L_2 = \{a^n b^m c^n m^n \mid m, n \geq k\}$, and $L_3 = \{ww \mid w \in \{a, b\}^i, i \geq k\}$, where k is a positive integer.

For the language $L_1 = \{a^n b^n c^n \mid n \geq 2\}$, we construct a CMLG $G = \langle V_T,\ \Lambda_D, \Lambda_H,\ 2,\ \Psi\rangle$ where $V_T = \{a, b, c\}$, $\Lambda_D = \{A, B, C, b, c\}$ and with the rules

$*(\ A\ )$	$(\ 1/\overline{C}\ )*$	$a(1/\overline{A}\ *\ 1/A\ b)$
$a(\ 1/\overline{A}\ *\ 1/B\ b)$	$b(\ 1/\bar{b}\ \ 1/\overline{B}\ *\ 2/B\ c)$	$b(\ 1/\bar{b}\ \ 2/\overline{B}\ *\ 2/C\ c)$
$c(\ 2/\bar{c}\ \ 2/\overline{C}\ *\ 2/C)$	$c(\ 2/\bar{c}\ \ 2/\overline{C}\ *\ 1/C)$	

For the language $L_2 = \{a^n b^m c^n d^m \mid m, n \geq 1\}$, we construct a CMLG $G = \langle V_T,\ \Lambda_D,\ \Lambda_H,\ 2,\ \Psi\rangle$ where $V_T = \{a, b, c, d\}$, $\Lambda_D = \{A, B, C, D, c, d\}$ and with the rules

$*(\ A\ )\quad (\ 1/\overline{D}\ )*$	$a(\ 1/\overline{A}\ *\ 1/A\ c)$	$b(\ 1/\overline{A}\ *\ 2/B\ d)$
$b(\ 2/\overline{B}\ *\ 2/B\ d)$	$b(\ 2/\overline{B}\ *\ 2/C\ d)$	$c(\ 1/\bar{c}\ 2/\overline{C}\ *\ 2/C)$
$c(\ 1/\bar{c}\ 2/\overline{C}\ *\ 2/D)$	$d(\ 2/\bar{d}\ 2/\overline{D}\ *\ 2/D)$	$d(\ 2/\bar{d}\ 2/\overline{D}\ *\ 1/D)$

For the language $L_3 = \{ww \mid w \in \{a, b\}^i, i \geq 2\}$, we construct a CMLG $G = \langle V_T,\ \Lambda_D,\ \Lambda_H,\ 2,\ \Psi\rangle$ where $V_T = \{a, b\}$, $\Lambda_D = \{U, V, Y, a, b\}$ and with the rules

$*(\ U\ )$	$(\ 1/\overline{Y}\ )*$	$a(\ 1/\overline{U}\ *\ 1/a\ U)$
$b(\ 1/\overline{U}\ *\ 1/b\ U)$	$0(\ 1/\bar{a}\ 1/\overline{U}\ *\ 2/V\ a)$	$0(\ 1/\bar{b}\ 1/\overline{U}\ *\ 2/V\ b)$
$0(\ 1/\bar{a}\ 2/\overline{V}\ *\ 2/V\ a)$	$0(\ 1/\bar{b}\ 2/\overline{V}\ *\ 2/V\ b)$	$a(\ 2/\bar{a}\ 2/\overline{V}\ *\ 2/V\ a)$
$b(\ 2/\bar{b}\ 2/\overline{V}\ *\ 2/V\ b)$	$a(\ 2/\bar{a}\ 2/\overline{V}\ *\ 1/Y\ a)$	$b(\ 2/\bar{b}\ 2/\overline{V}\ *\ 1/Y\ b)$

(iii) We have already shown that grammars in $G \in \mathbf{G}_c$, $c \geq 2$, can express non-context-free languages. Inclusion of all context-free languages is shown by reduction from the Hays-Gaifman dependency grammars. The rule set Π of these grammars contain rules of the following three types:

1. $*(X)$ — gives word categories the elements of which may govern the sentence,
2. $X(V_1\ V_2\ \ldots\ V_n\ *\ Y_1\ Y_2\ \ldots Y_m)$ — gives those categories which may derive directly from the category X and specified their relative positions, and
3. $X : w$ — gives for word category X a word w belonging to it.

We construct a CMLG $G = \langle V_T,\ \Lambda_D,\ \Lambda_H,\ c,\ \Psi\rangle$ where $V_T = \{w \mid (X : w)\ \in \Pi\}$, $\Lambda_D = \{X \mid X : w\ \in \Pi\}$ and with the set Ψ consisting of rules $(*(X))$ for each $(*(X))\ \in \Psi$, and of rules

$$w(\overline{X}\ 1/V_1\ 1/V_2\ \ldots\ 1/V_n\ *\ 1/Y_1\ Y_2\ \ldots Y_m),\ \text{and}$$

correct: $w(1/\overline{X}\ldots$

$$w(1/V_1\ 1/V_2\ \ldots\ 1/V_n\ *\ 1/Y_1\ Y_2\ \ldots Y_m\ \overline{X})$$

for each $(X(V_1\ V_2\ \ldots\ V_n\ *\ Y_1\ Y_2\ \ldots Y_m))\ \in \Pi$ and $(X : w)\ \in \Pi$.

(iv)Languages of CMLGs have linear growth property because CMLGs can be reduced to ERL-$\mathcal{S}^{\mathrm{c},\Gamma}_{\mathrm{pd}}$-Gs that have this property. $\square$

We denote the class of CMLG with c colors by $\mathbf{G}_c$.

CMLGs can associate to the input general dependency graphs. In fact, for any finite dependency graph attached to a string of word tokens, there is a CMLG that generates the string and associates this structure to the string:

Theorem 7 *If $D = (V, E)$ is a directed graphs without cycles of length 1 and $\prec$ is an order among the vertices V, then there is a CMLG that generates a string $v_1v_2\ldots v_{|V|} \in V^*$, where $v_i \prec v_{i+1}$ for all $1 \leq 1 \leq |V| - 1$, with a derivation tree whose storage operations encode the edges E.*

Proof. A edges (links) of the directed graph D can be colored in a correct manner using a method that has been informally presented in Yli-Jyrä (to appear). After the links have been colored, we know the number of colors required, and it is easy to extract from the colored D a CMLG that generates the string $v_1v_2\ldots v_{|V|}$ and associates the desired derivation tree for the string. The details are omitted for brevity. $\square$

However, the current definition of CMLGs has the restriction that the number of links leaving each word token is bounded by the grammar. Linguistically, having no free dependents is a severe restriction, but we believe that it is relatively easy to simulate such features in the CMLG.

Theorem 8 *Link grammars (Sleator and Temperley 1991) without connectors that can link one or more tokens are reducible to $G \in \boldsymbol{G}_1$.*

Proof. Omitted for brevity. $\square$

It should be noted, that underspecific link colors could be used in the rules of CMLGs in order to gain more flexibility when the possible linearisations are unknown. An underspecific rule can be seen as a rule-schema that is expanded to a finite set of normal rules.

11.6 c-Colored t-Non-projective Dependency Grammar

Linguistically oriented dependency grammars describe usually acyclic graphs. We are therefore interested to find a fragment of CMLG that would generate only acyclic graphs.

It is a well known that acyclicity of finite graphs is not first-order definable property. We get acyclicity neither as a by-product of derivation, because the derivation trees of the underlying non-terminal-free Norm$\mathcal{S}_{\mathrm{pd}}^{c,\Gamma}$-grammars can be completely different from the link structure represented by the storage operations. There is no limit for the length of "almost cyclic" paths that can be contained in the link structure. Thus, we have to enforce acyclicity of the link structure by some other means, by making a restriction to a subclass of acyclic structures.

In the following, we will propose a parametrized solution that is based on a new complexity measure called the *non-projectivity depth* of dependency paths.

Definition 9 A sequence of directed dependency links connecting string position i to string position j so that j is transitively governed by i is called a *dependency path.* The *non-projectivity depth* of an acyclic dependency path is the sum of the number of visited string positions for which the incoming (governor) link is shorter than the outgoing (dependent) link and the number of positions where the incoming link is stored to a pushdown whose index is greater than the index of the pushdown containing the outgoing link.

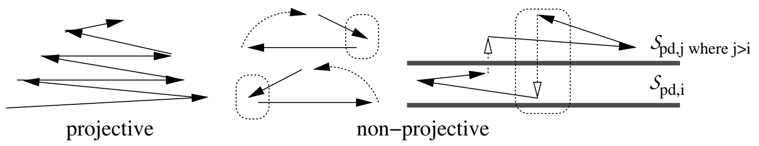
projective
non–projective
$\mathcal{S}_{\mathrm{pd},j}$ where j>i
$\mathcal{S}_{\mathrm{pd},i}$

FIGURE 3 Cycles contain at least one special kind of linkage (in rounded boxes) that are not found in *acyclic projective* graphs.

Using the non-projectivity depth is motivated by the facts that (i) it is always greater than zero for cyclic dependency paths, and (ii) it is constantly zero for projective dependency trees. Moreover, if we let grammars assign each token a the maximum non-projectivity depth of a path from an independent token to token a, the grammars will fail to build cyclic dependency paths, because such the maximum is not well-defined if a path is acyclic. Based on these observations, we define a subset of CMLGs that can generate only acyclic dependency graphs (but not all them).

Definition 10 A *colored t-non-projective dependency grammar (CNDG)* is a structure $G = \langle V_T, \Lambda_H, \Lambda_D, c, \Psi, t\rangle$ where V_T, Λ_D, Λ_H, c, and Ψ are defined in the same way as done for CMLGs, and t is the bound for non-projectivity depth in dependency paths.

The semantics of grammar G is much like in the case of CMLGs (Definition 7). However, there are the following differences:

- The stack alphabet will be $\Gamma = \{(\overleftarrow{\overline{x}}, i), (\overrightarrow{\overline{x}}, i) \mid x \in \Lambda_D, 0 \leq i \leq t\}$.
- The functions $\lambda, \rho : (\Lambda_D \cup \Lambda_H) \to 2^{\Gamma}$ are defined in such a way that

$$\begin{aligned} \lambda(\overline{x}) &= \{(\overrightarrow{\overline{x}}, i) \mid i \in [0..t]\} & \rho(\overline{x}) &= \{(\overleftarrow{\overline{x}}, i) \mid i \in [0..t]\} \\ \lambda(x) &= \{(\overleftarrow{\overline{x}}, i) \mid i \in [0..t]\} & \rho(x) &= \{(\overrightarrow{\overline{x}}, i) \mid i \in [0..t]\}. \end{aligned}$$

- From the obtained productions of the forms (11.5) and (11.6) we keep only those productions where the counters i in the pairs (X, i) (i) increase monotonically from incoming (governor) links to outgoing (dependent) links, (ii) increase strictly when an outgoing link is longer than an incoming link of the same side, (iii) increase strictly in left outgoing links with color p when there is a right incoming link with a color $q \geq p+1$, and (iv) do not increase more than necessary. This is expressed formally as follows:

  Let $(a_i, l_i) = v_i$ and $(b_j, r_j) = y_j$, where $1 \leq i \leq n$ and $1 \leq j \leq m$, be the stack symbols and $\alpha \in V_T \cup \{\epsilon\}$ be the lexical anchor in a constructed production

$$X\ \text{POP}_{p_n}(v_n) \cdots \text{POP}_{p_1}(v_1) \to \text{if TRUE then}\ \alpha X\ \text{PUSH}_q(y_m y_{m-1} \ldots y_1).$$

  This production is kept if, for all i, j, $1 \leq i \leq n$, $1 \leq j \leq m$, it holds that

$$\begin{aligned} a_i \in \overleftarrow{\Gamma} &\text{ implies } t_i = \max\{l_*, r_*, l_{n,i+1},\ s_{p_i}\}, \text{ and} \\ b_j \in \overrightarrow{\Gamma} &\text{ implies } u_j = \max\{l_*, r_*, r_{j-1,1}\}, \end{aligned}$$

erratum: swap $\overleftarrow{\Gamma}$ and $\overrightarrow{\Gamma}$

  where

$$\begin{aligned} l_* &= \max\{0\} \cup \{l_i & &\mid 1 \leq i \leq n,\ a_i \in \overrightarrow{\Gamma}\} \\ r_* &= \max\{0\} \cup \{r_j & &\mid 1 \leq j \leq m,\ a_j \in \overleftarrow{\Gamma}\} \\ l_{n,k} &= \max\{0\} \cup \{l_i + 1 & &\mid k \leq i \leq n,\ a_i \in \overrightarrow{\Gamma}\} \\ r_{k,1} &= \max\{0\} \cup \{r_j + 1 & &\mid 1 \leq j \leq k,\ a_j \in \overleftarrow{\Gamma}\} \\ s_p &= \max\{0\} \cup \{r_i + 1 & &\mid 1 \leq i \leq m,\ a_i \in \overleftarrow{\Gamma}, p < q\}. \end{aligned}$$

Theorem 9 *Classes of CNDG with at least two colors are mildly context-sensitive.*

Proof. The proof can be given in a similar way as in Theorem 6. Note in particular, that the example grammars for languages L_1, L_2 and L_3 and the representation for the Hays-Gaifman dependency grammars given there have bounded non-projectivity depth. □

The following theorem relates CNDGs to CMLGs:

Theorem 10 *Every CNDG can be reduced to a CMLG.*

Proof. Instead of constructing a nonterminal-free Norm$\mathcal{S}_{\mathrm{pd}}^{c,\Gamma}$-grammar directly from each CNDG as we did above, we will now have to construct in a similar way a CMLG where counters for non-projectivity depth are visible already in the link labels and in colored rules. □

The number of productions in CNDGs can much larger than in corresponding CMLGs. However, it is possible to parse the sentence first with a CMLG and then re-parse the parse forest using CNDGs that will filter out dependency graphs whose non-projectivity depth is greater than t. When used in this way, CNDGs may in fact provide more efficient filtering than what is generally possible by complete methods (e.g. backtracking search) for acyclicity testing.

11.7 Dependency Grammars for Trees

An acyclic dependency graph is a dependency tree if and only if

- it has a unique root
- all the word tokens except the root are governed by exactly one other node.

To obtain dependency grammars that assign dependency trees to the strings, we can specialize colored non-projective dependency grammars so that these two requirements are satisfied. First, we require that all the rules of the forms (11.9) and (11.10) contain exactly one governor link, and that the rules of the form (11.7) or the form (11.8) contain only one category ($n = 1$ or $m = 1$). Furthermore, the derivations where rules of both forms (11.7) and (11.8) must be discarded.

11.8 Conclusion

We have presented new classes of link and dependency grammars, namely the Colored Multiplanar Link Grammar (CMLG) and its subtypes, the c-Colored t-Non-projective Dependency Grammar (CNDG). The semantics of these grammars was given by reduction to Extended Right-Linear $\mathcal{S}_{\mathrm{pd}}^{c,\Gamma}$-grammar Wartena (2001), which immediately relates CMLGs and CNDGs with existing families of grammars.

The important contribution of this paper is to show that CMLGs and CNDGs are mildly context-sensitive and that the number of colors c available in CMLGs induce an infinite hierarchy of classes of CMLGs. Furthermore, a sub-hierarchy of CNDGs in each class of CMLGs is obtained by restricting the non-projectivity depth of the dependency paths. Dependency grammars that generate dependency trees up to a bounded non-projectivity depth form a subset of CNDGs.

We argue that the presented grammars have linguistic and practical relevance, because (i) the core ideas of the CMLG derivations have been tested against a small treebank (Yli-Jyrä 2003), (ii) the CMLG hierarchy provides a useful complexity measure for natural language sentences (iii) CMLGs are mildly context-sensitive and (iv) lexicalized, and (v) they give rise to finite-state approximations that assign non-projective dependency structures to strings (Yli-Jyrä 2004).

(This paper version appears in FGNancy 2004 pre-proceedings.)

Acknowledgements

The work was partially funded by NorFA under the personal Ph.D. scholarship (ref.nr. 010529) of the first author.

References

Castaño, J. M. 2003. Global index grammars and descriptive power. In R. T. Oehrle and J. Rogers, eds., *Proceedings of MOL 8, 2003*.

Gaifman, H. 1965. Dependency systems and phrase-structure systems. *Inf. Control* 8(3):304–37.

Gazdar, G. 1988. Applicability of indexed grammars to natural languages. In U. Reyle and C. Rohrer, eds., *Natural Language Parsing and Linguistic Theories*. Dordrecht: Reidel.

Hays, D. G. 1964. Dependency theory: A formalism and some observations. *Language* 40:511–525.

Joshi, A. K. 1985. Tree Adjoining Grammars: how much context-sensitivity is required to provide reasonable structural descriptions? In D. Dowty, L. Karttunen, and A. Zwicky, eds., *Natural Language Parsing*, pages 206–250. Cambridge: Cambridge University Press.

Sleator, Daniel and Davy Temperley. 1991. Parsing english with a link grammar. Technical Report CMU-CS-91-196, Carnegie Mellon University, Computer Science.

Tesnière, L. 1959. *Éléments de Syntaxe Structurale*. Paris: Editions Klincksieck.

Vijay-Shanker, K., David Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalism. In *25th ACL*, pages 104–111. Stanford, CA.

Wartena, C. 2001. Grammars with composite storages. In M. Moortgat, ed., *LACL'98*, vol. 2014 of *LNAI*, pages 266–285.

Weir, David J. 1994. Linear iterated pushdowns. *Computational Intelligence* 10(4):422–430.

Yli-Jyrä, A. 2003. Multiplanarity - a model for dependency structures in treebanks. In *The Second Workshop on Treebanks and Linguistic Theories*. Växjö, Sweden.

Yli-Jyrä, A. 2004. Axiomatization of non-projective dependency trees through finite-state constraints that analyse crossing bracketings (preliminary title). In *the COLING 2004 workshop "Recent Advances in Dependency Grammar"*. University of Geneva, Switzerland.

Yli-Jyrä, A. to appear. Coping with dependencies and word order or how to put Arthur's court into a castle. In H. Holmboe, ed., *Nordisk Sprogteknologi 2003. Årbog for Nordisk Sprogteknologisk Forskningsprogram 2000–2004*.