11

A Hierarchy of Mildly Context-Sensitive Dependency Grammars

ANSSI YLI-JYRÄ AND MATTI NYKÄNEN

Department of Modern Languages, University of Helsinki and School of Computing, University of Eastern Finland
Email: Anssi.Yli-Jyra@helsinki.fi, Matti.Nykanen@uef.fi

ABSTRACT. We generalize Link Grammars with crossing dependencies and projective Dependency Grammars with a *gradient projectivity condition* and multiple heads. The generalizations rely on right linear grammars that encode the links and dependencies via index storage instructions. The underlying machinery and the stepwise refinements of the definitions connect the generalized grammars to Linear Indexed Grammars and the mildly context sensitive hierarchy of Wartena's right linear grammars that use composite index storages. The gradient projectivity measure – the *non-projectivty depth* – relies on a normalized storage type whose runs represent particular multiplanar decompositions for dependency graphs. This decomposition does not aim at minimal graph coloring, but focuses, instead, to rigidly necessary color changes in right linear processing. Thanks to this multiplanar decomposition, every non-projectivity depth k characterizes strictly larger sets of *acyclic dependency graphs and trees* than the planar projective dependency graphs and trees.

11.1 Introduction

In this paper, we generalize Link Grammars (Sleator and Temperley 1993) and projective Dependency Grammars (Hays 1964, Gaifman 1965) with crossing dependencies and then characterize the generalizations using formal language theory.

In terms of Dependency Grammars (DGs) (Tesnière 1959), word-order is distinct from the dependency tree that analyzes the structure of the sentence. However, Hays (1964) and Gaifman (1965) have

Proceedings of Formal Grammar 2004.
Paola Monachesi, Gerhard Jäger,
Gerald Penn, and Shuly Wintner (eds.).
Copyright © 2014, CSLI Publications.

formalized Tesnière's ideas so that their DGs describe only *projective* linearisations. There is, however, a need for a mildly context-sensitive (MCS) superclass of the Hays-Gaifman DGs that would capture also non-projective dependencies, e.g. scrambling, the possibility of the elements of a sentence to lie in arbitrary permutations. The notion of mild context-sensitivity is an attempt by Joshi (1985) to characterize the formal properties of grammars that are needed to describe the semantically coherent structures of natural language. However, it seems that many grammars capturing unrestricted scrambling fail to be MCS (Joshi 1985), *cf.* Global Index Grammar (GIG) (Castaño 2003).

Limited scrambling is 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 generalizing the index pushdown with an *index storage* of type $\mathcal{S}$.

We base our investigations on Extended Right Linear $\mathcal{S}^c_{\text{pd}}$-Grammars (ERL-$\mathcal{S}$-Gs) whose storage type is a tuple of c pushdowns that are seen as one pushdown during writing (Wartena 2001). Some new restrictions on ERL-$\mathcal{S}^c_{\text{pd}}$-Gs are developed to obtain various classes of DGs. The obtained DGs are used to describe non-projective dependencies and scrambling. They include also the Hays-Gaifman DGs (Hays 1964, Gaifman 1965) and Link Grammars (Sleator and Temperley 1993) as their subclasses. The main contributions of this paper are:

- to formalize an unambiguous multiplanar decomposition method (Yli-Jyrä 2003, 2004b) using composite storages;
- to generalize the projectivity condition as the *non-projectivity depth*
- to show that there are hierarchies of non-projective DGs and Link Grammars generating MCS languages;
- to propose further research on storages and right linear processing.

The paper is structured as follows. Section 11.2 defines basic storage (refined in Sections 11.3 and 11.4) and grammar types (refined in 11.5 and 11.7). Section 11.6 studies some central formal properties of *Colored Multiplanar Link Grammars.* Section 11.8 concludes the paper.

11.2 The Basic Machinery

11.2.1 Storage Type

In this section, we define an abstract storage type $\mathcal{S}$ from which we refine storage types $\mathcal{S}_{\text{triv}}$, $\mathcal{S}_{\text{pd}}$, $\mathcal{S}^{c,\Gamma}_{\text{pd}}$, $N(\mathcal{S}^{c,\Gamma}_{\text{pd}})$ (in Section 11.3), and

$R(N(\mathcal{S}_{\text{pd}}^{c,\Gamma}))$ (in Section 11.4). The aim of the stepwise refinements of the storage type is to relate our proposal to prior work.

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 $m(\pi)\colon C \to \{\text{TRUE}, \text{FALSE}\}$, and to each $\phi \in \Phi$ the corresponding partial function $m(\phi)\colon C \to C$.

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

11.2.2 Concatenating Storage Type

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 with respect to writing is denoted as $\circ_w$.

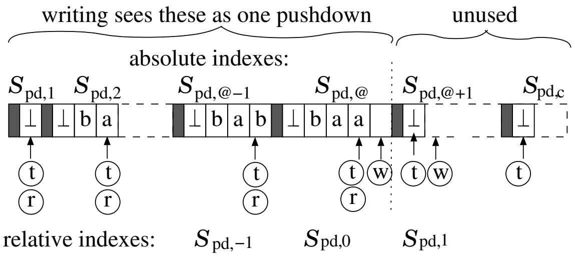

FIGURE 1 Storage type $(...(((\mathcal{S}_{\text{triv}} \circ_w \mathcal{S}_{\text{pd},1}) \circ_w \mathcal{S}_{\text{pd},2}) \circ_w \mathcal{S}_{\text{pd},3}) \dots) \circ_w \mathcal{S}_{\text{pd},c}$ with applicable operations T (t), POP (r) and PUSH (w)

We restrict our attention to storages of type $(...(((\mathcal{S}_{\text{triv}} \circ_w \mathcal{S}_{\text{pd},1}) \circ_w \mathcal{S}_{\text{pd},2}) \circ_w \mathcal{S}_{\text{pd},3}) \dots) \circ_w \mathcal{S}_{\text{pd},c}$ that is intuitively a tuple $\langle \mathcal{S}_{\text{pd},1}, \mathcal{S}_{\text{pd},2}, \mathcal{S}_{\text{pd},3}, \dots, \mathcal{S}_{\text{pd},c}\rangle$ 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 $\mathcal{S}_{\text{pd},p+1}, \dots, \mathcal{S}_{\text{pd},c}$ are empty. Otherwise each pushdown can be used independently of the others.

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

- the configurations are $C = (\bot(\Gamma \cup \{\sharp\})^{*})^{c}$, where $\bot \notin \Gamma$ is a special symbol denoting the bottom of the pushdown, and $\sharp$ is a semaphore reserved for implementing a restriction in Section 11.3;

- the unique initial and final configuration is $C_\mathrm{i} = C_\mathrm{f} = \{\bot\}^c$;
- the predicates are $\Pi = \{\textsc{t}_p(a) \mid 1 \le p \le c, a \in \Gamma \cup \{\bot, \sharp\}\}$;
- the instructions are $\Phi = \{\textsc{id}, \textsc{undef}\} \cup \{\textsc{push}_p(\beta) \mid 1 \le p \le c,$ $\beta \in (\Gamma \cup \{\sharp\})^+\} \cup \{\textsc{pop}_p(\beta) \mid 1 \le p \le c, \beta \in (\Gamma \cup \{\sharp\})^+\}$.

The basic meanings of predicates Π and instructions Φ are given by

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

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

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 an Extended Domain of Locality*, EDL-CFL-$\mathcal{S}$-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

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

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

The set $\mathbf{S} = ((V_N \times C) \cup V_T)^*$ is called the set of *sentential forms.* A sentential form $\sigma = \alpha\,(X, \kappa)\,\beta \in \mathbf{S}$ is said to *derive* a sentential form $\tau = \alpha\gamma\beta \in \mathbf{S}$, written $\sigma \Rightarrow \tau$, for some $\alpha, \beta, \gamma \in \mathbf{S}$, if

- P contains some production r of the form (11.1), $m(\pi)(\kappa) = \textsc{true}$, and $\gamma = w$, or,
- P contains some production r of the form (11.2), $m(\pi)(m(\phi_1)(\kappa)) = \textsc{true}$, $m(\phi_1\phi_2)(\kappa) \in C$, 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 Z by (Z, κ_0).

The *language* generated by the grammar G is defined as $L(G) = \{a_1 \ldots a_n \mid ((S, \kappa_0)) \Rightarrow^* \tau, \tau \in V_T^*\}$, where $\Rightarrow^*$ denotes the reflexive and transitive closure of the derivation relation $\Rightarrow$.

The following definition and theorems relate some restrictions of the class of EDL-CFL-$\mathcal{S}$-Gs. The languages generated by a class $\mathcal{C}$ of grammars is denoted by $\mathcal{L}(\mathcal{C})$.

Definition 4 A *Right Linear $\mathcal{S}$-Grammar with an Extended Domain of Locality*, EDL-RL-$\mathcal{S}$-G, is a EDL-CFL-$\mathcal{S}$-G whose productions of the form (11.2) are such that $\zeta_1 \in V_T^*$ and $\zeta_2 \in \epsilon$. A *Right Linear $\mathcal{S}$-Grammar*, RL-$\mathcal{S}$-G, is an EDL-RL-$\mathcal{S}$-G whose productions of the form (11.2) are such that $\phi_1 = \textsc{id}$ and $\phi_2 \in \Phi$.

Every EDL-CFL-$\mathcal{S}$-Grammar whose productions of the form (11.2), are such that $\phi_1 = \textsc{id}$ and $\phi_2 \in \Phi$, is a *Context-Free Linear $\mathcal{S}$-Grammar*, CFL-$\mathcal{S}$-G, (Weir 1994).

Every CFL-$\mathcal{S}$-G whose productions of the form (11.2) are such that $\zeta_1 \in V_N$ and $\zeta_2 \in V_T \cup \{\epsilon\}$, is an *Extended Right Linear $\mathcal{S}$-Grammar*, ERL-$\mathcal{S}$-G, (Wartena 2001).[1]

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

Proof. The inclusions $\mathcal{L}(\mathcal{C}_{\text{EDL-CFL-}\mathcal{S}\text{-G}}) \subseteq \mathcal{L}(\mathcal{C}_{\text{CFL-}\mathcal{S}\text{-G}})$ and $\mathcal{L}(\mathcal{C}_{\text{EDL-RL-}\mathcal{S}\text{-G}}) \subseteq \mathcal{L}(\mathcal{C}_{\text{RL-}\mathcal{S}\text{-G}})$ follow from the definition of the grammars. To show that the reverse inclusions hold, we replace productions of the form (11.2) by expanding them syntactically into

$$X[\textsc{id}] \rightarrow \text{if}\ \ \textsc{true}\ \ \text{then}\ \ \zeta_1\, Q^Y_{\phi_1\pi\phi_2}[\textsc{id}]\,\zeta_2$$

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

$$\begin{aligned} Q^Y_\epsilon[\textsc{id}] &\rightarrow \text{if}\ \ \textsc{true}\ \ \text{then}\ Y\,[\textsc{id}] \\ Q^Y_{\pi'\omega}[\textsc{id}] &\rightarrow \text{if}\qquad \pi'\ \text{then}\ Q^Y_\omega\,[\textsc{id}] \\ Q^Y_{\phi'\omega}[\textsc{id}] &\rightarrow \text{if}\ \ \textsc{true}\ \ \text{then}\ Q^Y_\omega\,[\phi'] \end{aligned}$$

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 (11.2) in the expanded grammar contain only productions where $\phi_1 = \textsc{id}$ and $\phi_2 \in \Phi$. The expanded grammar recognizes the language of the original grammar. $\square$

Theorem 2 *For every RL-$\mathcal{S}$-G there is an ERL-$\mathcal{S}$-G generating the same language and the same storage instruction sequences for strings.*

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

[1]Although ERL-$\mathcal{S}$-Gs have left-linear productions such as $X \rightarrow Y[\phi_2]\ a$, the missing right linear production $X \rightarrow a\ Y[\phi_2]$ can be simulated easily via productions $X \rightarrow \zeta_1\ Y[\phi_2]$ and $\zeta_1 \rightarrow a$ where $\zeta_1 \in V_N$. The attribute "right linear" signals that the storage instruction $[\phi_2]$ goes with Y rather than ζ_1

11.3 Normalized Storage

11.3.1 The Intuition

If the number of pushdowns is larger than the number of crossings in the dependency structure that is associated with the terminal string, EDL-CFL-$\mathcal{S}_{\mathrm{pd}}^{c,\Gamma}$-Gs have freedom to allocate different pushdowns for different stack symbols.

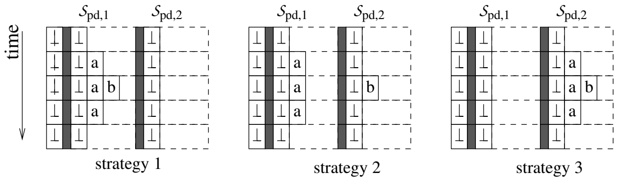

FIGURE 2 An example of unmotivated ambiguity in storage allocation

In order to eliminate strategies 2 and 3 shown in Figure 2, the core restrictions, (1) the LIFO discipline and (2) the concatenating storage type, are complemented with conditions (3) and (4) that say that writing to an empty pushdown $\mathcal{S}_{\mathrm{pd},p}$, $p \geq 2$, is allowed only if

(3) $\mathcal{S}_{\mathrm{pd},p-1}$ has a stack alphabet symbol on it, and

(4) the written symbol stays longer than a symbol previously written to pushdown $\mathcal{S}_{\mathrm{pd},p-1}$.

Let Φ' be an extended set of instructions on the *Normalized* storage type $N(\mathcal{S}_{\mathrm{pd}}^{c,\Gamma})$. 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). The primitive instructions PUSH and POP are restricted to private use.

11.3.2 The Implementation

The meanings of the new instructions are given using the primitives PUSH and POP in combination with an auxiliary semaphore symbol $\sharp$ that signals a need for a crossing read operation (Figure 3):

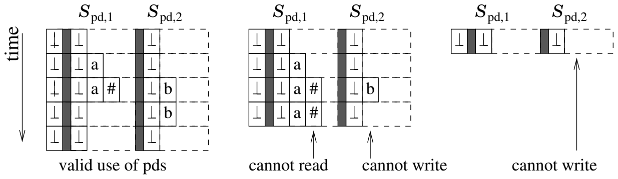

FIGURE 3 Only a crossing link occupies a new pushdown without a failure

$$m(\textsc{rpush}_p(\beta))(\kappa) = \begin{cases} m(\textsc{push}_p(\beta))(\kappa), & \text{if } p = 1 \vee \neg m(\textsc{t}_p(\bot))(\kappa); \\ m(\textsc{push}_{p-1}(\sharp)\,\textsc{push}_p(\beta))(\kappa), & \text{if } p \neq 1 \wedge m(\textsc{t}_p(\bot) \wedge \textsc{t}_{p-1}(\Gamma))(\kappa), \end{cases}$$

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

$$\begin{cases} m(\textsc{pop}_p(a))(\kappa), & \text{if } p = c \vee \neg m(\textsc{t}_p(\sharp))(\kappa); \\ m(\textsc{pop}_p(\sharp)\,\textsc{pop}_p(a))(\kappa), & \text{if } p \neq c \wedge m(\textsc{t}_p(\sharp) \wedge \textsc{t}_{p+1}(\Gamma))(\kappa). \end{cases}$$

To write to an empty pushdown $\mathcal{S}_{\mathrm{pd},p}$, $p > 1$, using instruction $\textsc{rpush}_p(\beta)$, pushdown $\mathcal{S}_{\mathrm{pd},p-1}$ must have a stack symbol as its topmost element. This implements condition (3).

When an empty pushdown $\mathcal{S}_{\mathrm{pd},p}$, $p > 1$, is written with instruction $\textsc{rpush}_p(\beta)$, the semaphore symbol $\sharp$ is inserted into pushdown $\mathcal{S}_{\mathrm{pd},p-1}$. This semaphore is removed only when $\mathcal{S}_{\mathrm{pd},p-1}$ is read while pushdown $\mathcal{S}_{\mathrm{pd},p}$ is still nonempty. If the higher pushdown $\mathcal{S}_{\mathrm{pd},p}$ is read too early, the derivation is blocked reflecting the condition (4).

Theorem 3 *EDL-CFL-N($\mathcal{S}_{\mathrm{pd}}^{c,\Gamma}$)-Gs using* RPUSH *and* RPOP *instructions can be reduced to EDL-CFL-$\mathcal{S}_{\mathrm{pd}}^{c,\Gamma}$-Gs that do not use these instructions.*

Proof. The new instructions can be regarded as abbreviations extending the transformation in the proof of Theorem 1 as follows:

Nonterminals $Q^Y_{\textsc{rpush}_1(\beta)\omega}$ and $Q^Y_{\textsc{rpop}_c(\beta)\omega}$ are replaced, respectively, with nonterminals $Q^Y_{\textsc{push}_1(\beta)\omega}$ and $Q^Y_{\textsc{pop}_c(\beta)\omega}$. Nonterminals $Q^Y_{\textsc{rpush}_p(\beta)\omega}$, where $p \neq 1$, create the grammar rules

$$Q^Y_{\textsc{rpush}_p(\beta)\omega}[\textsc{id}] \to \text{if } \neg\, \textsc{t}_p(\bot) \text{ then } Q^Y_\omega[\textsc{push}_p(\beta)],$$
$$Q^Y_{\textsc{rpush}_p(\beta)\omega}[\textsc{id}] \to \text{if } \textsc{t}_p(\bot) \wedge \textsc{t}_{p-1}(\Gamma) \text{ then } Q^Y_\omega[\textsc{push}_{p-1}(\sharp)\,\textsc{push}_p(\beta)],$$

and nonterminals $Q^Y_{\textsc{rpop}_p(\alpha)\omega}$, where $p \neq c$, create the grammar rules

$$Q^Y_{\textsc{rpop}_p(\alpha)\omega}[\textsc{id}] \to \text{if } \neg\, \textsc{t}_p(\sharp) \text{ then } Q^Y_\omega[\textsc{pop}_p(\alpha)],$$
$$Q^Y_{\textsc{rpop}_p(\alpha)\omega}[\textsc{id}] \to \text{if } \textsc{t}_p(\sharp) \wedge \textsc{t}_{p+1}(\Gamma) \text{ then } Q^Y_\omega[\textsc{pop}_p(\sharp\alpha)]. \qquad \square$$

Definition 5 A *Context-Free Linear Normalized-$\mathcal{S}_{\mathrm{pd}}^{c,\Gamma}$-Grammar with an Extended Domain of Locality*, EDL-CFL-$N(\mathcal{S}_{\mathrm{pd}}^{c,\Gamma})$-G, is a EDL-CFL-$\mathcal{S}_{\mathrm{pd}}^{c,\Gamma}$-Grammar whose rules do not directly use PUSH and POP instructions, but use the RPUSH and RPOP instructions instead.

Proposition 4 *EDL-CFL-N($\mathcal{S}_{\mathrm{pd}}^{c,\Gamma}$)-Gs allocate pushdowns for different stack symbols so that they conform the storage restrictions (1) - (4) in Section 11.3.1.*

Definition 6 A *Nonterminal-Free* $N(\mathcal{S}_{\text{pd}}^{c,\Gamma})$*-Grammar* is a EDL-RL-$N(\mathcal{S}_{\text{pd}}^{c,\Gamma})$-G $G = (V_N,\ V_T,\ P,\ S,\ \kappa_0)$ for which $V_N = \{S, X\}$ and whose productions are of the following forms:

$$\begin{array}{rlll} S[\textsc{id}] \to \text{if} & \textsc{true} & \text{then}\ X[\phi_{2,1}] & (11.3)\\ X[\phi_{1,r}] \to \text{if} & \wedge_{1\le p\le c}\, \textsc{t}_p(\bot) & \text{then}\ \epsilon & (11.4)\\ X[\phi_{1,r}] \to \text{if} & \pi & \text{then}\ aX[\phi_{2,1}] & (11.5)\\ X[\phi_{1,r}] \to \text{if} & \pi & \text{then}\ X[\phi_{2,r+1}] & (11.6) \end{array}$$

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

The sentential forms $\langle(S,\kappa_0)\rangle$ and $\alpha(X, \beta\bot^n)$ of EDL-RL-$N(\mathcal{S}_{\text{pd}}^{c,\Gamma})$-G are abbreviated as S and $\alpha\beta$, respectively.

11.4 Relatively Indexed Tuple of Pushdowns

The smallest index of a pushdown that is succeeded only by empty pushdowns in a given configuration κ is denoted by $@(\kappa)$. We do not want to refer to pushdowns by absolute numbers $1, ..., c$, but by relative numbers $1-c, 2-c, ..., 0, 1$ where 0 corresponds to the absolute index $@(\kappa)$.

Definition 7 Let $\mathcal{S}_{\text{pd}}^{c,\Gamma} = (C,\ C_{\text{i}},\ C_{\text{f}},\ \Phi,\ \Pi,\ m)$ be a concatenating storage type. Then storage type $R(\mathcal{S}_{\text{pd}}^{c,\Gamma}) = (C, C_{\text{i}}, C_{\text{f}}, \Phi, \{\textsc{empty}\}, m')$ is a *Relative-*$\mathcal{S}_{\text{pd}}^{c,\Gamma}$, $R(\mathcal{S}_{\text{pd}}^{c,\Gamma})$. For this storage type, the basic meanings of predicates Π and instructions Φ are given by

$$\begin{aligned} m'(\textsc{id})(\kappa) &= m(\textsc{id})(\kappa);\\ m'(\textsc{empty})(\kappa) &= m(\wedge_{1\le p\le c}\, \textsc{t}_p(\bot))(\kappa);\\ m'(\textsc{ins}'_p(\beta))(\kappa) &= \begin{cases} m(\textsc{ins}_{(p'+@(\kappa))}(\beta))(\kappa) & \text{if } 1 \le (p' + @(\kappa)) \le c;\\ \text{undefined} & \text{otherwise;} \end{cases} \end{aligned}$$

where $\textsc{ins} \in \{\textsc{push}, \textsc{pop}, \textsc{rpush}, \textsc{rpop}\}$ and $@(\kappa)$ is the largest $1 \le i \le c$ for which $m(\textsc{t}_i(\bot)) = \textsc{false}$ or $i = 1$.

11.5 Colored Multiplanar Link Grammar (CMLG)

In some experiments with a multiplanar decomposition of dependency trees (Yli-Jyrä 2003), a very good coverage of dependency trees was obtained with only a small tuple of pushdowns that follow the intuition of $N(\mathcal{S}_{\text{pd}}^{c,\Gamma})$. In this section, we formalize a multiplanar grammar using this normalized storage type.

Definition 8 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 *head labels*, c is the *number of colors*, and Ψ is the set of *colored rules* of the forms

$$*(0/Y_1 \dots Y_m) \tag{11.7}$$

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

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

$$\epsilon(p_1/V_1 \dots p_n/V_n \ * \ q/Y_1 \dots Y_m) \tag{11.10}$$

where $a \in V_T$, $0 \notin V_T$, and $V_1, V_2, \dots, V_n, Y_1, Y_2, \dots Y_m \in \Lambda_D \cup \Lambda_H$, $q \in \{0, 1\}$, and $p_1, p_2, \dots, p_n \in \{1-c, \dots, 0\}$, with $p_i \leq p_{i+1}$ for $1 \leq i \leq n$. The uses of the rule type (11.10) are restricted in such a way that, (i) $p_i < q$ for some $1 \leq i \leq n$ and (ii) for all $1 \leq i \leq n$ and $1 \leq j \leq m$, either $p_i = q$ or $(Y_j, V_i) \notin F^+$, where F^+ is the transitive closure of $F = \{(V_i, Y_j) \mid \alpha(p_1/V_1 \dots p_n/V_n * q/Y_1 \dots Y_m) \in \Psi, 1 \leq i \leq n, 1 \leq j \leq m\}$.

The semantics of the grammar G is defined by reducing it to a Nonterminal-Free $R(N(\mathcal{S}_{\text{pd}}^{c,\Gamma}))$-Grammar $G = (V_N, V_T, P, S, \kappa_0)$, with stack alphabet $\Gamma = \{x, x \mid x \in \Lambda_D\}$ and set P containing the production

$$S[\text{ID}] \rightarrow \text{if } \ \text{TRUE} \ \text{ then } X[\text{RPUSH}_1(y_m \dots y_1)]$$

for each rule of type (11.7) and the production

$$X[\text{RPOP}_{p_n}(v_n) \cdots \text{RPOP}_{p_1}(v_1)] \rightarrow \text{if } \ \text{EMPTY} \ \text{ then } \epsilon$$

for each rule of type (11.8); and respectively the productions

$$X[\text{RPOP}_{p_n}(v_n) \cdots \text{RPOP}_{p_1}(v_1)] \rightarrow \text{if TRUE then } \ aX[\text{RPUSH}_q(y_m \dots y_1)],$$
$$X[\text{RPOP}_{p_n}(v_n) \cdots \text{RPOP}_{p_1}(v_1)] \rightarrow \text{if TRUE then } \quad X[\text{RPUSH}_q(y_m \dots y_1)],$$

where $v_i \in \lambda(V_i)$ and $y_i \in \rho(Y_i)$ for each rule of type (11.9) and (11.10) and functions $\lambda, \rho : (\Lambda_D \cup \Lambda_H) \rightarrow 2^\Gamma$ that are given by

$$\lambda(\overline{x}) = \lambda(x) = \{x\} \qquad\qquad \rho(\overline{x}) = \rho(x) = \{x\}.$$

To make the rule syntax more elegant, any colored link $0/V$, where $V \in V_N$, will be written simply as V.

11.6 Mild Context-Sensitivity of CMLGs

Definition 9 A class $\mathcal{C}$ of grammars is *MCS* if the grammars of this class are (i) polynomial-time parseable, (ii) they capture multiple dependencies, limited crossing dependencies and the copy language, and if the grammars in $\mathcal{C}$ generate (iii) a proper superclass of context-free

TABLE 1 An overview of the hierarchy of $\mathcal{L}(\mathcal{C}_{\mathrm{CMLG}(c)})$, $c \in \{0, 1, 2, ...\}$

$\mathcal{L}(\mathcal{C}_{\mathrm{CMLG}(0)})$	$=$	$\emptyset$		
$\mathcal{L}(\mathcal{C}_{\mathrm{CMLG}(1)})$	$=$	$\mathcal{L}(\mathcal{C}_{\mathrm{ERL}\text{-}\mathcal{S}_{\mathrm{pd},1}\text{-G}}) = \mathcal{L}(\mathcal{C}_{\mathrm{CFG}})$		
$\mathcal{L}(\mathcal{C}_{\mathrm{CMLG}(2)})$	$\subseteq$	$\mathcal{L}(\mathcal{C}_{\mathrm{ERL}\text{-}\mathcal{S}_{\mathrm{pd},2}\text{-G}}) = \mathcal{L}(\mathcal{C}_{\mathrm{ERLIG}})$	$\subseteq$	$\mathcal{L}(\mathcal{C}_{\mathrm{LIG}})$
$\mathcal{L}(\mathcal{C}_{\mathrm{CMLG}(3)})$	$\subseteq$	$\mathcal{L}(\mathcal{C}_{\mathrm{ERL}\text{-}\mathcal{S}_{\mathrm{pd},3}\text{-G}})$		
...	...	...		

languages where (iv) all the languages L have the linear growth property: for every sufficiently long string $w \in L$ there is another string $w' \in L$ which is at most k symbols shorter.

The class of CMLG with c colors is denoted by $\mathcal{C}_{\mathrm{CMLG}(c)}$. The language classes $\mathcal{L}(\mathcal{C}_{\mathrm{CMLG}(0)})$, $\mathcal{L}(\mathcal{C}_{\mathrm{CMLG}(1)})$, $\mathcal{L}(\mathcal{C}_{\mathrm{CMLG}(2)})$, ... form a hierarchy (Figure 1). Its levels are strictly contained to the levels of Wartena's (2001) hierarchy whose level 2 contains the languages generated by Extended Right Linear Indexed Grammars (ibid.).

Theorem 5 *Any $\mathcal{C}_{CMLG(c)}$, $c \geq 2$, is mildly context-sensitive.*

Proof. We establish the properties (i) - (iv) for $\mathcal{C}_{\mathrm{CMLG}(c)}$ as follows:

(i) Every CMLG $G \in \mathcal{C}_{\mathrm{CMLG}(c)}$ reduces to a polynomially parseable grammar ERL-$\mathcal{S}_{\mathrm{pd}}^{c,\Gamma}$-G that simulates the storage sequences of G and produces enough structural descriptions to parse G.

(ii) The capability to describe multiple dependencies, limited crossing dependencies and copying is usually demonstrated using the 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 $L_1 = \{a^n b^n c^n \mid n \geq 1\}$, we construct a CMLG $G = \langle V_T, \Lambda_D, \Lambda_H, 2, \Psi \rangle$ with $V_T = \{a, b, c\}$, $\Lambda_D = \{A, B, C, b, c\}$, and the rules Ψ:

  $$\begin{array}{llll} *(\,A\,) & (\,C\,)* & a(A * A\, b) & b(\,b\,A * B\,c) \\ b(\,b\;\;A * 1/B\;c) & b(\,-1/b\;\;B * B\;c) & c(\,c\;\;B * C) & c(\,c\;\;C * C). \end{array}$$

  E.g., string 'aabbcc' is derived by $S \Rightarrow \bot A \Rightarrow a \bot bA \Rightarrow aa \bot bbA \Rightarrow aab \bot b\sharp \bot cB \Rightarrow aabb \bot \bot ccB \Rightarrow aabbc \bot \bot cC \Rightarrow aabbcc \bot \bot C \Rightarrow aabbcc.$

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

  $$\begin{array}{llll} *(\,A\,) & (\,D\,)* & a(\,A * A\;c) & a(\,A * B\;c) \\ b(\,B * 1/B\;d) & b(\,B * 1/C\;d) & b(\,B * C\;d) & \\ c(\,-1/c\;C * C) & c(\,-1/c\;C * D) & d(\,d\;D * D). & \end{array}$$

E.g., string 'abbcdd' is derived by $S \Rightarrow \perp A \Rightarrow a \perp cB \Rightarrow ab \perp c\sharp \perp dB \Rightarrow abb \perp c\sharp \perp ddC \Rightarrow abbc \perp \perp ddD \Rightarrow abbcd \perp \perp dD \Rightarrow abbcdd \perp \perp D \Rightarrow abbcdd$.

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

$*(\ T\ )$	$(\ V\ )*$	$a(\ T\ *\ T\ a)$	$b(\ T\ *\ T\ b)$
$\epsilon(\ a\ T\ *\ 1/U\ a')$	$\epsilon(\ b\ T\ *\ 1/U\ b')$	$\epsilon(\ -1/a\ U\ *\ U\ a')$	$\epsilon(\ -1/b\ U\ *\ U\ b')$
$a(\ U\ a'\ *\ V\ )$	$b(\ U\ b'\ *\ V\ )$	$a(\ V\ a'\ *\ V\ )$	$b(\ V\ b'\ *\ V\ )$

  E.g., string 'abab' is derived by $S \Rightarrow \perp T \Rightarrow a \perp aT \Rightarrow ab \perp abT \Rightarrow ab \perp a\sharp \perp b'U \Rightarrow ab \perp \perp b'a'U \Rightarrow aba \perp \perp b'V \Rightarrow abab \perp \perp V \Rightarrow abab$.

(iii) The inclusion of all context-free languages to $\mathcal{C}_{\mathrm{CMLG}(c)}$ is shown by reduction from the Hays-Gaifman DGs. The rules in the set P are of the following three types:

1. $*(X)$ — gives word categories the elements of which may govern the sentence,
2. $X(V_1 \ldots V_n\ *\ Y_1 \ldots Y_m)$ — 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 $V_T = \{w \mid (X : w)\ \in P\}$, $\Lambda_D = \{X \mid X : w\ \in P\}$ and with the set Ψ consisting of rules $(*(X))$ for each $(*(X))\ \in \Psi$, and of rules

$$w(X\ V_1 \ldots V_n\ *\ Y_1 \ldots Y_m), \text{ and } w(V_1 \ldots V_n\ *\ Y_1 \ldots Y_m\ X),$$

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

(iv) Languages $\mathcal{L}(\mathcal{C}_{\mathrm{CMLG}(c)})$ have the linear growth property because CMLGs reduce to ERL-$\mathcal{S}_{\mathrm{pd}}^{c,\Gamma}$-Gs whose languages have the property. □

Let w be a string of terminal symbols $w = v_1 v_2 \ldots v_n$ and $H = (V, S, E)$ be a graph with word vertices $V = \{v_i \mid 1 \leq i \leq n\}$, successor edges $S = \{(i, i+1) \mid 1 \leq i < n\}$, and other dependency edges $E \subseteq V \times V$ describing the link structure of w.

Theorem 6 *There is a CMLG that generates exactly the string w and processes its link structure H.*

Proof. The successor edges and other edges in the link structure graph H are labeled uniquely to avoid overgeneralization. The corresponding pushdowns are associated with the links using a multiplanar decomposition method Yli-Jyrä (2004b) that simulates the behavior of the normalized storage in Section 11.3. After the pushdowns have been allocated, we can extract the rules for a CMLG that generates the string w and associates the desired derivation tree for it. □

Theorem 7 *Link Grammars (Sleator and Temperley 1993) are reducible to grammars in* $\mathcal{C}_{CMLG(1)}$.

Proof. The lexicon of a Link Grammar uses a minus sign ($-$) to indicate left links and plus ($+$) to indicate right links. For example, a transitive verb 'painted' would have a lexical entry

$$\textit{painted} : S- \;\&\; O+.$$

This lexical entry corresponds to a CMLG rule *painted*$(S * O)$. The multi-connectors in a Link Grammars are eliminated by creating chaining lexical entries. Elaborating the details of the reduction is left to the reader. □

11.7 Colored Non-projective Dependency Grammar

11.7.1 The Intuition

Definition 10 *The projectivity condition:* If word A depends immediately on word B and some word C intervenes them, then C depends transitively on A or B (Marcus 1967).

A dependency graph G is acyclic and projective if it is 1-planar, loop-free and its dependency paths follow the writing ordering of pushdowns (Yli-Jyrä 2005). However, the projectivity condition applies also to multiplanar dependency graphs: If the multiplanar decomposition is based on the normalized storage, the non-projectivity coincides with similar local features that can be observed in planar graphs: If any cyclic graph in Figure 4 is stripped from its non-projective features by removing some incident links, only a projective acyclic graph would remain.

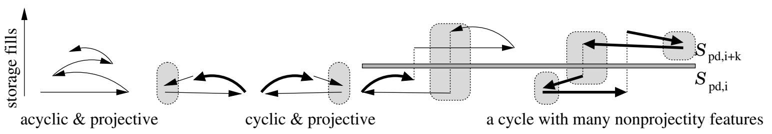

FIGURE 4 Cycles involve a non-projective feature: the path descends towards the bottom of the pushdown or the concatenating storage

A multiplanar graph is acyclic if the non-projectivity depth is finite for all dependency paths. In contrast, the non-projectivity depths of the *projective* cycles in Figure 4 are undefined. However, if one thick link is removed from any of these cyclic graphs, the remaining acyclic graph would be non-projective and have a nonzero, finite non-projectivity depth.

Definition 11 The *non-projectivity depth* of an acyclic dependency path is the sum of (i) the number of string positions for which the incoming link is shorter than the outgoing link and (ii) the number of string positions where the incoming link is stored to a pushdown whose index is greater than the index of the pushdown containing the outgoing link.

11.7.2 The Implementation

Directed (dependency) graphs can be produced by CMLGs that distinguish between incoming (dependent) and outgoing ($\overline{\text{head}}$) links. We will now define a subset of CMLGs that can generate acyclic c-colored dependency graphs up to a given non-projectivity depth.

Definition 12 A *Colored 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 8). However, there are the following differences:

- An extended stack alphabet $\Gamma = \{(\overleftarrow{x}, i), (\overrightarrow{x}, i) \mid x \in \Lambda_D, 0 \le i \le t\}$ indicates link directions and the non-projectivity depth.
- The functions $\lambda, \rho : (\Lambda_D \cup \Lambda_H) \to 2^\Gamma$ are given by

$$\begin{array}{ll} \lambda(\overline{x}) = \{(\overrightarrow{x}, 0), \ldots, (\overrightarrow{x}, t)\} & \rho(\overline{x}) = \{(\overleftarrow{x}, 0), \ldots, (\overleftarrow{x}, t)\} \\ \lambda(x) = \{(\overleftarrow{x}, 0), \ldots, (\overleftarrow{x}, t)\} & \rho(x) = \{(\overrightarrow{x}, 0), \ldots, (\overrightarrow{x}, t)\}. \end{array}$$

- From the obtained subproductions of the forms (11.5) and (11.6) we keep only those where the counters i in the pairs (X, i) (i) increase monotonically from incoming ($\overline{\text{head}}$) links to outgoing (dependent) links, (ii) increase strictly when an outgoing link is longer than an incoming link on the same side, (iii) increase strictly in left outgoing links with color p when there is a right incoming link with a color $q \ge p + 1$, and (iv) do not increase more than necessary.

To describe the selection of the subproductions more precisely, let $(a_i, l_i) = v_i$, $1 \le i \le n$, and $(b_j, r_j) = y_j$, $1 \le j \le m$, be the stack symbols and let $\alpha \in V_T \cup \{\epsilon\}$ be the lexical anchor in a constructed production

$$X[\textsc{rpop}_{p_n}(v_n) \cdots \textsc{rpop}_{p_1}(v_1)] \to \text{if } \textsc{true} \text{ then } \alpha X[\textsc{rpush}_q(y_m \cdots y_1)]. \quad (11.11)$$

Subproduction (11.11) is kept if the depth counter of every left outgoing link $v_i = (\overleftarrow{x}_i, l_i)$ and the depth counter of every right outgoing link $y_j = (\overrightarrow{x}_j, r_j)$ satisfies the contraints

$$l_i = \max\{0, g, 1{+}h_{i+1}, 1{+}s_i\} \text{ and } r_j = \max\{0, g, 1{+}h_{j-1}\},$$

where g is the maximum non-projectivity depth of the rule's incoming links among $v_1, \ldots, v_n, y_1, \ldots, y_m$, and integers h_{i+1} and h_{j-1} are, respectively, the maximum non-projectivity depths of such left and right incoming links that are shorter than the links v_i and y_j (if there is none, the default depth is -1), and s_i is the maximum non-projectivity depth (defaulting to -1) of right incoming links that have a higher color than v_i.[2]

Theorem 8 *Every CNDG can be reduced to a CMLG.*

Proof. Instead of constructing a Nonterminal-Free $R(N(\mathcal{S}_{\text{pd}}^{c,\Gamma}))$-Grammar directly from each CNDG as we did above with CMLGs, we construct a CMLG where counters for non-projectivity depth are visible already in the stack alphabet Γ. □

Theorem 9 *Classes of CNDG with at least two colors are MCS.*

Proof. The proof can be given in a similar way as in Theorem 11.7.2. Note in particular, that the links in the grammars for languages L_1, L_2 and L_3 and the representation for the Hays-Gaifman DGs in Theorem can be directed in such a way that the non-projectivity depth of the generated dependency structures is 0. □

If no word is allowed to have more than one head (Section 11.7.3), it suffices to have one subproduction for each non-projectivity depth. Moreover, the bound for the non-projectivity depth in natural language treebanks is likely to be very close to 1.

11.7.3 Restriction to Dependency Trees

DGs that assign dependency trees to the strings are obtained from CNDGs via three refinements: (i) rules of the forms (11.9) and (11.10) are restricted to contain exactly one dominating link, (ii) rules of the forms (11.7) are restricted to contain exactly one dominating word, and (ii) rules of the forms (11.8) are restricted to contain no dominating link.

[2]To capture more structures with a very low t, we could split the constant TRUE in productions into two complementary cases. The first case, where the storage is tested empty after the POP's, would not increase counters $a_1, \ldots, a_n$ on the basis of $b_1, \ldots, b_m$ and vise versa.

11.8 Conclusion and Discussion

We have presented some generalizations of Link Grammars and projective DGs, namely Colored Multiplanar Link Grammar (CMLG) and its variant, Colored Non-projective Dependency Grammar (CNDG). The semantics of these grammars was given by reduction to Extended Right Linear $\mathcal{S}^{c,\Gamma}_{\mathrm{pd}}$-Grammar Wartena (2001), which immediately relates CMLGs and CNDGs with existing families of grammars. There remain questions concerning the properties of our grammar hierarchy. However, the presented ideas can be fruitful because:

- The multiplanar coloring is a simple way to represent and parse dependency treebanks and measure the complexity of dependency trees (Yli-Jyrä 2003, Gómez-Rodríguez and Nivre 2013);
- CMLGs are lexicalized and mildly context-sensitive;
- CNDGs can be approximated with FSMs (Yli-Jyrä 2004a);

CMLGs could be made even more powerful by generalizing the storage type. In particular, we could define concatenating storages where there is no limit for the number of new pushdowns, but all but the rightmost c pushdowns must be empty at any given time. This requires a modification to the meaning of POP:

$$m(\textsc{pop}_p(\beta))(\langle\alpha_1,\cdots,\alpha_{p-1},\alpha_p\beta^{\mathrm{r}},\alpha_{p+1},\cdots,\alpha_c\rangle)=\begin{cases}\langle\alpha_2,\cdots,\alpha_c,\bot\rangle & \text{if } \alpha_1=\bot;\\ \langle\alpha_1,\cdots,\alpha_c\rangle & \text{if } \alpha_1\neq\bot.\end{cases}$$

The formal properties of this generalization are still open. It is no surprise that the safety of the rules type (11.10) will need to be established to avoid undecidability.

The method of Section 11.7 designed for DAGs can be compared with a simple method (Yli-Jyrä 2012) that verifies the acyclicity and connectivity of undirected multiplanar trees.

Acknowledgements

The work has been funded by the Nordic Academy for Advanced Study, (ref. no. 010529) and the Academy of Finland (ref. no. 270354).

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.

Gómez-Rodríguez, Carlos and Joakim Nivre. 2013. Divisible transition systems and multiplanar dependency parsing. *Computational Linguistics* 39(4):799–845.

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.

Marcus, Solomon. 1967. *Algebraic Linguistics; Analytical Models*, vol. 29 of *Mathematics in Science and Engineering*, chap. VI (Subordination and Projectivity), pages 200–246. New York and London: Academic Press.

Sleator, Daniel and Davy Temperley. 1993. Parsing English with a link grammar. In *3rd International Workshop on Parsing Technologies*, pages 277–291.

Tesnière, L. 1959. *Éléments de Syntaxe Structurale*. Paris: 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. 2004a. Axiomatization of restricted non-projective dependency trees through finite-state constraints that analyse crossing bracketings. In G.-J. M. Kruijff and D. Duchier, eds., *Proc. of the Workshop of Recent Advances in Dependency Grammar*, pages 33–40. Geneva, Switzerland.

Yli-Jyrä, A. 2004b. 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*, pages 123–137. Copenhagen: Museum Tusculanums Forlag.

Yli-Jyrä, Anssi. 2005. Approximating dependency grammars through intersection of star-free regular languages. *International Journal of Foundations of Computer Science* 16(3).

Yli-Jyrä, Anssi. 2012. On dependency analysis via contractions and weighted FSTs. In D. Santos, K. Lindén, and W. Nganga, eds., *Shall we Play the Festschrift Game? Essays on the Occasion of Lauri Carlson's 60th Birthday*. Berlin: Springer-Verlag.