Axiomatization of Restricted Non-Projective Dependency Trees through Finite-State Constraints that Analyse Crossing Bracketings

Anssi YLI-JYRÄ
Department of General Linguistics, P.O. Box 9, FIN-00014 University of Helsinki
anssi.yli-jyra@helsinki.fi

Abstract

In this paper, a representation for syntactic dependency trees (D-trees) is defined through a finite set of axioms. The axiomatized representation constitutes a string that can encode non-projective D-trees of restricted structural complexity. Upper-bounds for the structural complexity of these D-trees are fixed through the following new parameters: *proper embracement depth* d, *nested crossing depth* c, and *non-projectivity depth* t.

In the representation, syntactic dependencies between words are indicated with pairs of brackets. When the brackets indicate dependencies that cross each other, the crossing pairs of brackets are distinguished by assigning separate colors to each of them. These colors are allocated in a way (Yli-Jyrä and Nykänen, 2004) that ensures a unique representation for each D-tree, and entails that languages whose *nested crossing depth* is not bounded cannot be captured using a fixed number of colors.

Although the axiomatization is finite, it ensures that the represented dependency structures are trees. This is possible because the described D-trees have bounded *non-projectivity depth*. The axioms are also regular because *proper embracement depth* of represented D-trees is bounded.

Our representation suggests that extra strong generative power can be squeezed out of finite-state equivalent grammars. Bracketed D-tree representations (*cf.* annotated sentences) are structural descriptions that are assigned to their subsequences (*cf.* generated strings or yields of trees) where brackets and other special-purpose characters have been omitted.

1 Introduction

Recently, many dependency syntactic parsers using finite-state machines (FSMs) have been presented (Kahane et al., 1998; Elworthy, 2000; Nasr et al., 2002; Oflazer, 2003; Yli-Jyrä, 2004a). This article shows that a finite-state equivalent grammatical system is capable of assigning even non-projective syntactic dependency trees — or their representations — to terminal strings. An appropriate representation is conveniently defined through a set of axioms presented in this work. The complexity of the structures assigned is bounded by some special parameters.

1.1 Motivation

We argue that the possibilities of FSMs have not been fully exploited in dependency syntax. So far, almost all the dependency parsers that use FSMs take them merely as subroutines in a system whose generative power exceeds regular languages. Although there are some *pure* finite-state approaches to surface syntactic parsing (Krauwer and des Tombe, 1981; Abney, 1996; Koskenniemi, 1997; Yli-Jyrä, 2004a) there seems to be a lack of a pure finite-state approach that is capable of assigning non-projective dependency structures to the input strings.

The applicability of finite-state systems to natural language syntax has been questioned since Chomsky (1957), who suggested that center embedding in natural language is unbounded. In contrast to this view, a recent corpus-based study (Karlsson, in print) suggests an opposite generalisation according to which there is an absolute limit $(2 – 3)$ on center-embedding of subordinate clauses in matrix clauses, on top of which there are also category restrictions on which type of center-embeddings are allowed at each embedding level. Although such limits may lack some mathematical elegancy, they may entail some other kind of mathematical beauty (*e.g.* the closure properties of regular languages) and, moreover, new possibilities in the framework of parameterized complexity can become available. In natural language engineering, where ambiguity generated by a syntactic parser can be very high, limits on syntactic complexity may resolve some ambiguity and reduce the number of non-typical analyses generated by the parser. Several such limits on the complexity of D-trees are proposed in this paper.

To facilitate implementation of a non-projective

dependency grammar with FSMs, this paper introduces a suitable string representation. This representation is inspired by Colored Non-Projective Dependency Grammar (Yli-Jyrä and Nykänen, 2004), where multiple index pushdowns are used to store symbols for dependency links. The axiomatization deals with crossing dependencies and enforces acyclicity of the represented dependency graph. It involves several extensions that are not present in earlier encoding schemes where dependencies are also indicated through matching pairs of symbols (Oflazer, 2003; Yli-Jyrä, 2004a).

1.2 Non-Projective Dependency Trees

In dependency syntax, the analysis of a sentence is given as a dependency tree (D-tree) whose nodes correspond — as assumed in this paper — to the words of the analyzed sentence. A D-tree consists of dependency links (directed arcs) drawn above the sentence — and implicitly, of the sentence itself. The D-tree shows which words are related to which words and in what way. Figure 1 gives an example of a D-tree.

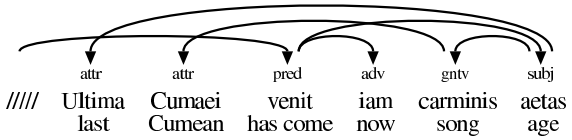

Figure 1: This tree analyses the Latin sentence "Ultima Cumaei venit iam carminis aetas" (Vergil: Eclogues IV.4) that means "The last era of the Cumean song has now arrived". The analysis is adapted from Covington (1990). We added the sentence-initial node (/////) and the arc labels.

The graphical representation for the D-tree is interpreted as follows. If $v_1 \rightarrowtail v_2$ is a (directed) arc between two nodes, we will say that v_2 *depends immediately on* v_1 (or, conversely, v_1 *governs* v_2 *immediately*), and that v_2 is an *immediate syntactic dependent* of v_1 (and v_1 is the *immediate syntactic governor* of v_2). (Mel'čuk 1988.)

In each D-tree, there is a unique non-governed node. For this purpose, we have reserved an external wall node (/////) that is placed on the left of the sentence, while the words in the sentence are always governed by exactly one node. A D-tree is *non-projective*, if some of the arcs cross each other when the tree is drawn above the sentence.

1.3 The New Representation

The string representation (Figure 2) for a D-tree consists of overlapping views — string subsequences — that realize different aspects of the D-tree encoding. For example, substrings delimited by a pair or boundary symbols ($^{\#}$) are called *nodes* in the string representation. The first node corresponds to the wall (/////) and the other nodes contain single *word tokens* of the sentence:

$$^{\#}\ ///// \ ^{\#}\ \text{Ultima}\ ^{\#}\ \text{Cumaei}\ ^{\#}\ \text{venit}\ ^{\#}\ \text{iam}\ ^{\#}\ \text{carminis}\ ^{\#}\ \text{aetas}\ ^{\#}$$

For any given D-tree there is a unique way to assign colors to its arcs. When the coloring is done, the result is a *colored D-tree*. At each node of the colored D-tree, there is a unique *active* color i that is used for the arcs that connect the node to the right. In the string representation this is indicated by special tokens $:i:$, $i:$ as follows:

$$^{\#}\ 1{:}\ ^{\#}\ 2{:}\ ^{\#}\ {:}2{:}\ ^{\#}\ 3{:}\ ^{\#}\ {:}3{:}\ ^{\#}\ {:}3{:}\ ^{\#}\ 1{:}\ ^{\#}$$

The arcs are encoded with separate pairs of brackets as follows:

$$\begin{array}{cccccccccccccc}
^{\#} & [_1 & ^{\#} & & ^{\#} &]_1 & & ^{\#} & & ^{\#} & & ^{\#} & & ^{\#} \\
^{\#} & & ^{\#} & [_2 & ^{\#} & & & ^{\#} & & ^{\#} & & ^{\#} &]_2 & ^{\#} \\
^{\#} & & ^{\#} & & ^{\#} & [_2 & ^{\#} & & ^{\#} &]_2 & & ^{\#} & & ^{\#} \\
^{\#} & & ^{\#} & & ^{\#} & & [_3 & ^{\#} & & ^{\#} &]_3 & & ^{\#} & \\
^{\#} & & ^{\#} & & ^{\#} & & [_3 & ^{\#} & \rangle_3 & ^{\#} & & ^{\#} & & ^{\#} \\
^{\#} & & ^{\#} & & ^{\#} & & & ^{\#} & & \langle_3 & ^{\#} &]_3 & & ^{\#}
\end{array}$$

Each substring $[_p \cdots_p]_p$, $[_p \cdots_p \rangle_p$, and $\langle_p \cdots_p]_p$, where the intervening string $\cdots_p$ has a balanced bracketing w.r.t. $[_p,]_p$-brackets, corresponds to an arc between two nodes in the D-tree. When these subsequences are put on top of each other we obtain the following combination:

$$^{\#}\ [_1\ ^{\#}\ [_2\ ^{\#}\ [_2\ ^{\#}\ ]_1\ [_3\ ^{\#}\ \rangle_3\ ^{\#}\ ]_2\ \langle_3\ ^{\#}\ ]_3\ ]_2\ ^{\#}$$

For each arc $v_1 \rightarrowtail v_2$ in Figure 1, there is a label x that tells *how* v_2 depends on v_1. These labels are attached to the brackets as follows:

$$^{\#}\ [_1\ \text{pred}\ ^{\#}\ [_2\ \text{attr}\ ^{\#}\ [_2\ \text{attr}\ ^{\#}\ \text{pred}\ ]_1\ [_3\ \text{subj adv}\ ^{\#}\ \text{adv}\ \rangle_3\ ^{\#}\ \text{attr}\ ]_2\ \langle_3\ \text{gntv}\ ^{\#}\ \text{gntv subj}\ ]_3\ \text{attr}\ ]_2\ ^{\#}.$$

The direction of each arc is indicated by adding an over-line to the dependent-side labels:

$$[_2\ \overline{\text{attr}}\ [_2\ \overline{\text{attr}}\ \overline{\text{pred}}\ ]_1\ \overline{\text{adv}}\ \rangle_3\ \langle_3\ \overline{\text{gntv}}\ \overline{\text{subj}}\ ]_3.$$

The non-projectivity depth of an arc measures its specially defined distance from the wall node /////. In each node, the non-projectivity depth of an outgoing arc is greater (or equal) than the depth of the incoming arc. The depth of arcs is indicated as follows:

$$^{\#}\ [_1\ 0\ ^{\#}\ [_2\ 1\ ^{\#}\ [_2\ 0\ ^{\#}\ 0\ ]_1\ [_3\ 0\ 0\ ^{\#}\ 0\ \rangle_3\ ^{\#}\ 0\ ]_2\ \langle_3\ 0\ ^{\#}\ 0\ 0\ ]_3\ 1\ ]_2\ ^{\#}.$$

When we combine all these views, we obtain a string that represents the D-tree of Figure 1. This string is shown in Figure 2.

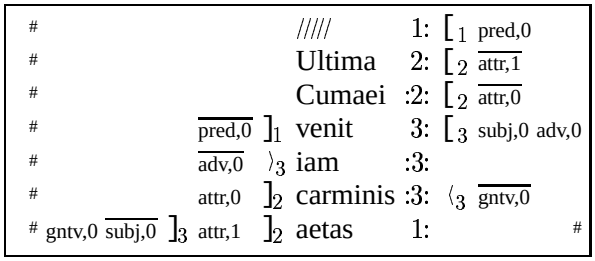

Figure 2: A string representation for a D-tree.

2 Prerequisites

We are going to define the string representation for D-trees with axioms that are given as extended regular expressions. The intersection of the languages described by these axioms is the set of valid representations. In the following, we define the alphabets and regular expressions used in the axioms.

Assume that Λ is the set of category labels for the arcs, and that d, c, and t are the parameters specifying upper bounds respectively for the *proper bracketing depth*, *nested crossing depth*, and *non-projectivity depth*. These depth measures will be precisely defined in an appropriate context.

2.1 Alphabets

In Figure 2, several different kinds of symbols are involved. They belong to the following alphabets:

- the sets B_L, B_R, A_L, A_R consisting respectively of brackets $[_i$, $]_i$, $\langle_i$, $\rangle_i$, $1 \le i \le c$,
- the color selectors $E = \{\, i:\, ,\ :i:\ \mid 1 \le i \le c\}$,
- the dependent labels $L_D = \{\ (x,i) \mid x \in \Lambda, 0 \le i \le t\}$,
- the governor labels $L_H = \{\, \overline{x} \mid x \in L_D\}$,
- the set of word tokens W, and
- the set of special symbols $\{^{\#}, ////\}$.

The union of these alphabets is denoted as Σ. The union of the alphabets L_D and L_H is L.

2.2 Regular Expressions

Let A and B be sets of strings, and n a positive integer. In the axioms we will use the following regular operations: Kleene's closure (A^*), finite iteration (A^n), concatenation (AB), asymmetric difference ($A - B$), union ($A \cup B$), and intersection ($A \cap B$) — with this precedence order. The semantics of these operations is defined in the usual way. Parenthesis ($()$) is used for grouping. The boxed dot $\boxdot$ denotes the language $(\Sigma - \{^{\#}\})^*$.

A *context restriction of a center $\mathcal{X}$ in contexts* $\mathfrak{C}_1, \mathfrak{C}_2, \cdots, \mathfrak{C}_n$ is a regular operation where $\mathcal{X}$ is a subset of Σ^* and each context $\mathfrak{C}_i$, $1 \le i \le n$, is of the form $\mathcal{V}_i \_\_ \mathcal{Y}_i$, where $\mathcal{V}_i, \mathcal{Y}_i \subseteq \Sigma^*$. The operation is expressed using a notation

$$\mathcal{X} \Rightarrow \mathcal{V}_1 \_\_ \mathcal{Y}_1, \mathcal{V}_2 \_\_ \mathcal{Y}_2, \ldots, \mathcal{V}_n \_\_ \mathcal{Y}_n$$

and it defines the set of all strings $w \in \Sigma^*$ such that, for every possible $v, y \in \Sigma^*$ and $x \in \mathcal{X}$, for which $w = vxy$, there exists some context $\mathcal{V}_i \_\_ \mathcal{Y}_i$, $1 \le i \le n$, where both $v \in \Sigma^* \mathcal{V}_i$ and $y \in \mathcal{Y}_i \Sigma^*$.

The axioms in this paper produce a set of quite small automata and the satisfiability and usability of this lazy finite-state system for choosing a representation has been tested with real D-trees. (After these tests, the axioms presented here have undergone some editing that hopefully have not introduced typos or bugs.)

2.3 The Basic Structure

Axiom 1. (a) The string begins and ends with a node boundary ($^{\#}$). (b) Between each two boundaries there exists at least one word token or the wall $w \in W \cup \{////\}$. (c) Two tokens $w, w' \in W \cup \{////\}$ are always separated by a node boundary.

Axiom 2. (a) There are no two similar square brackets in a node. (b) The color indices i of closing brackets increase monotonically when we move from a right bracket towards the closest node boundary ($^{\#}$) on the left.

Axiom 3. (a) All the labels $x \in L$ belong to some surrounding bracket. (b) Each left (right) bracket has some label that is attached to it.

Axiom 4. (a) Angle brackets $A_L \cup A_R$ do not have more than one label attached to them. (b) Within each node, no angle bracket $\langle_i$ ($\rangle_i$) occurs inside a square bracket $[_i$ ($]_i$) having the same color i.

Axiom 5. (a) The wall ($////$) and all the word tokens $w \in W$ occur before a color selector $e \in E$. (b) All the color selectors $e \in E$ occur after a word token or the wall.

Axiom 6. (a) There is one and only one ungoverned node. (b) All other nodes are governed by some node. (c) No node depends immediately on more than one other node. (d) No node depends on itself.

These axioms are presented more formally as the following regular expressions:

$$^{\#}\Sigma^*{}^{\#} \tag{1a}$$

$$\Sigma^* - \Sigma^*{}^{\#}\,(\Sigma - (W \cup \{////\}))^*\ {}^{\#}\,\Sigma^* \tag{1b}$$

$$\Sigma^* - \Sigma^*\ (W \cup \{////\}) \boxdot (W \cup \{////\})\ \Sigma^*. \tag{1c}$$

$$\Sigma^* - \Sigma^*([_i \boxdot [_i \cup]_i \boxdot]_i)\Sigma^* \tag{2a}$$

$$\{]_i, \rangle_i\} \Rightarrow (\{^{\#}\} \cup B')L^* \_\_, \tag{2b}$$

$$L \Rightarrow (B_L \cup A_L)L^* \_\_,\ \_\_ L^*(B_R \cup A_R) \tag{3a}$$

$$((B_L \cup A_L) \Rightarrow \_\_ L) \cap ((B_R \cup A_R) \Rightarrow L \_\_) \tag{3b}$$

$$\Sigma^* - \Sigma^* A_L LL \Sigma^* - \Sigma^* LL A_R \Sigma^* \quad (4a)$$

$$\Sigma^* - \Sigma^* \rangle_i \boxdot]_i \Sigma^* - \Sigma^* [_i \boxdot \langle_i \Sigma^* \quad (4b)$$

$$(W \cup \{////\}) \Rightarrow \_\_ E \quad (5a)$$

$$E \Rightarrow (W \cup \{////\}) \_\_ \quad (5b)$$

$$\Sigma^* ////\Sigma^* - \Sigma^* ////\Sigma^* ////\Sigma^* \quad (6a)$$

$$W \Rightarrow L_H \boxdot \_\_, \_\_ \boxdot L_H \quad (6b)$$

$$\Sigma^* - \Sigma^* L_H \boxdot L_H \Sigma^* \quad (6c)$$

$$\Sigma^* - \Sigma^* B_L \boxdot B_R \Sigma^* \quad (6d)$$

where $1 \leq i \leq c, B' = \{]_i, \cdots,]_c, \rangle_i, \cdots, \rangle_c\}$

3 Proper Embracement Depth

In the context of D-trees, a counterpart notion for center-embedding of constituent trees is needed. We say that an arc $u_1 \rightarrowtail v_1$ *properly embraces* another arc $u_2 \rightarrowtail v_2$, if and only if $\min\{u_1, v_1\} \prec \min\{u_2, v_2\}$ and $\max\{u_2, v_2\} \prec \max\{u_1, v_1\}$, where $\prec$ is the linear precedence order among the nodes. The *proper embracement depth* d of a colored D-tree is the maximum number d of arcs $u_1 \rightarrowtail v_1$, $u_2 \rightarrowtail v_2$, ..., $u_d \rightarrowtail v_d$ where all the arcs have the same color and each $u_i \rightarrowtail v_i$, $1 \leq i \leq d-1$, properly embraces $u_{i+1} \rightarrowtail v_{i+1}$. This measure is applied to a D-tree in the Figure 3. Note that proper embracement does not generally imply that the arcs belong to a common path: in Figure 1 the arc aetas$\rightarrowtail$Ultima embraces properly the arc carminis$\rightarrowtail$Cumaei.

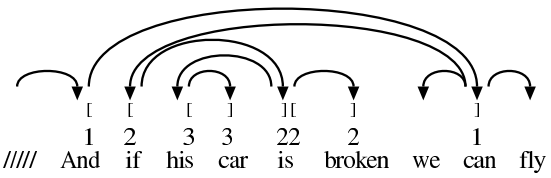

Figure 3: The proper embracement depth of this clause is 3.

An arc $u_1 \rightarrowtail v_1$ that shares a node with another arc $u_2 \rightarrowtail v_2$ does not properly embrace the later. If they overlap each other, the shorter of these arcs will be represented with a pair of an angle bracket and a square bracket as shown in Figure 2, unless the longer arc is already presented by an angle bracket. Thus, the maximum number of nested square brackets needed corresponds to the proper embracement depth of the colored D-tree.

In our representation, the *proper embracement depth* of trees is bounded by a fixed parameter d. This allows defining finite-state constraints that define bracketings up to a bounded number of nested square brackets. In the following, we will give axioms that check that brackets for each color are balanced and do not exceed the *proper embracement depth* d.

We define first, for each color $i \in [1, 2, 3, \ldots, c]$, a string homomorphism $g_i : \Sigma^* \rightarrow \{[_i,]_i\}^*$ in such a way that it essentially deletes all the other symbols except the brackets $[_i$ and $]_i$, and the regular languages $D_{1,d,i} \subseteq \{[_i,]_i\}^*$, $\Delta_{d,i} \in \Sigma^*$ in the following way:

$$D_{1,d,i} = \begin{cases} \epsilon & \text{if } d = 0; \\ (D_{1,d-1,i} \cup ([_i D_{1,d-1,i}]_i))^* & \text{if } d > 0. \end{cases}$$

$$\Delta_{d,i} = g_i^{-1}(D_{1,d,i}).$$

Some auxiliary languages are defined as follows:

$$\Delta'_{d,i} = \Delta_{d,i} - \Delta_{d,i} \langle_i \Delta_{d,i}$$

$$\Delta''_{d,i} = \Delta_{d,i} - \Delta_{d,i} \rangle_i \Delta_{d,i}.$$

Axiom 7. The $[_i$, $]_i$-bracketings must be balanced and the number of nested brackets is bounded.

Axiom 8. (a+b) Left (right) angle brackets match with a square bracket. (c) The arcs indicated with angle-square bracket pairs do not cross each other as in $[_i \ldots_i \langle_i \cdots_i \rangle_i \cdots_i]_i$.

These axioms are given more formally as follows:

$$\Delta_{d,i} \quad (7)$$

$$\langle_i \Rightarrow \_\_ \Delta_{d,i}]_i \quad (8a)$$

$$\rangle_i \Rightarrow [_i \Delta_{d,i} \_\_ \quad (8b)$$

$$\Sigma^* - \Sigma^* \langle_i \Delta_{d,i} \rangle_i \Sigma^* \quad (8b)$$

where $1 \leq i \leq c$

4 Nested Crossing Depth

We added colors to brackets because crossing brackets have to be separated by some means. Unfortunately, assigning colors to brackets entails new problems:

1. We can represent non-projective trees that are not typical for natural language. In particular, we conjecture that although we bound the number of colors c available, there is a set of colored trees (in the limit $d \rightarrow \infty$) that gives structural descriptions for the Bach language (*cf.* Joshi 1985), the strings of which consist of an equal number of a's, b's and c's.
2. In parsing, the colors must be selected in one way or in another and this results normally into an ambiguity where there are many colorings available. Thus, we need a discipline that tells how to assign colors to the arcs in an unambiguous way.

The first problem could be addressed *e.g.* by combining a constituent-based structure (topological fields etc.) with the dependency syntax. In our representation for D-trees, we need however a solution that addresses both of these problems. Such

a solution has been developed recently and presented in many ways: by means of constraints (Yli-Jyrä, 2003a), through an informal algorithm (Yli-Jyrä, 2004b), and very formally as a special index storage type used in Colored Non-projective Dependency Grammar (Yli-Jyrä and Nykänen, 2004). We conjecture, however, that there is no essential differences in the allocation disciplines defined in these works. In the following, we will adapt the constraint-based definition (Yli-Jyrä, 2003a) to the allocation of colors of brackets:

Axiom 9 (Plane locking). If a bracket $[_j$ is still open at a string position, the position cannot contain a bracket $[_i$ for which $i < j$.

An effect of this axiom can be seen in Figure 2, where a new color 3 is selected after *venit* although the color 1 is no more in use. The reason is that there is a bracket $[_2$ that is still open at that position.

Axiom 10 (Left conjoin). All the opening brackets belonging to the same node have the same color.

This axiom corresponds intuitively to the fact that there is no need to give different colors to arcs that do not cross each other.

Axiom 11 (Continuous tiling). A position cannot contain colored bracket $[_j$, where $1 < j < c$, if, on the left, there are *no* other brackets $[_j$ (of the same color) that remain opened at the position, except if there is, on the left, another bracket $[_i$ with $i = j-1$ (of the preceding color) that remains open at this position but will be matched with with a bracket $\rangle_i$ or $]_i$ that occurs before the bracket $[_j$ of the current position is closed with $]_j$.

This axiom corresponds to the fact that when a new color is introduced at some position (Figure 2), this is done due to a danger of having crossing brackets with the same color.

The actual effect of these three axioms is that for each D-tree there remains a unique way to assign colors, square brackets, and angle brackets to the arcs. The *nested crossing depth* c of a D-tree is the number of colors in a colored D-tree that conforms Axioms 9 – 11. In our representation, the nested crossing depth (*i.e.* the number of colors) is bounded.

Bounded nested crossing depth has considerable linguistic relevance. The length of the longest chain ($\frown\!\!\frown \cdots \frown$) of crossing edges is a *lower bound* for the nested crossing depth, but such chains are typically very short in natural language sentences. The possible upper bound for the nested crossing depth has been studied experimentally (Yli-Jyrä, 2003a; Yli-Jyrä, 2004b) with the result that in non-projective D-trees of some 700 Danish sentences, the number of required colors is pretty low (2 – 3). A few interesting exceptions[1] actually contained a chain of up to five crossing dependencies. Such complex examples seem to be successful combinations of non-local dependencies, and it may be very difficult to generalize what is possible and what is not. Nevertheless, we conjecture that D-trees of the Bach (or MIX) language (cf. Joshi 1985) are not captured in our system when the number colors is fixed, because Colored Non-Projective Dependency Grammar (Yli-Jyrä and Nykänen, 2004) is a linear context-free rewriting system.

In order to facilitate formalization of Axiom 11, we use of color selectors and the following axiom:

Axiom 12. (a) Color selector $:j:$, where $1 < j \leq c$, indicates that there is a left bracket $[_j$ that has not yet been closed. (b) Color selector $j:$ indicates that no bracket $[_j$ is open at that position.

If we assume also a bound for the proper embracement depth, we can present the above axioms more formally as follows:

$$[_j \Rightarrow \_\_ (\Delta_{d,j} - \Sigma^*\{[_i, \langle_i \mid 1 \leq i < j\}\Sigma^*)]_j \quad (9)$$

$$[_k \Rightarrow \_\_ \{L, [_k, \langle_k\}^* \ ^{\#} \quad (10)$$

$$j{:}\Rightarrow \_\_ (\Delta_{d,j\text{-}1}\{]_{j\text{-}1}, \rangle_{j\text{-}1}\}\,\Sigma^*\Sigma \cap [_j\Delta_{d,j}]_j) \quad (11)$$

$$:k{:} \Rightarrow [_k\Delta_{d,k}\_\_ \quad (12a)$$

$$\Sigma^* - (\Sigma^* - \Delta_{d,k})\ k{:}\ \ \Sigma^* \quad (12b)$$

where $1 \leq i < j \leq c, 1 \leq k \leq c$

5 Subcategorization

We have seen in Section 3 that angle brackets are used when several overlapping arcs share a common node. This corresponds to use of reduced bracketing for initial and final embedding in some systems (Krauwer and des Tombe, 1981; Yli-Jyrä, 2003b), and it facilitates linguistically appropriate bracketing with FSMs.

Our axiomatization (Section 6 in particular) requires that information about the labels and directions on the arcs of the node are locally present both in the dependent and the governor nodes. Unfortunately, this kind of duplication of the labeling information cannot be captured with regular axioms unless there is a limit on the amount of information that is duplicated. A solution would be to assign each square bracket an unsaturated subcategorization frame with a symbol that indicates a state in a special subcategorization automaton. The automata could be simulated by propagating — by means of

[1] *e.g. "Det har*$_1$ *både*$_2$ *noget*$_{1,3}$ *med*$_4$ *stolene*$_5$ *og*$_2$ *bordet at*$_3$ *gøre*$_4$ *— og*$_5$ *pladsen udenom."*

declarative constraints — the state information of each square bracket to the first angle bracket, and then further from one angle bracket to another. We have chosen, however, a more restricted approach for brevity, although we do not argue that it is the most elegant and general solution. This approach is presented in the sequel.

Figure 4: Angle brackets are used for non-proper embraced bracketing. The additional labels of the square bracket correspond to the labels attached to angle brackets.

We assume that the number of left or right arcs per color is bounded by an integer r. Thus, at most r labels can be associated with one square bracket. The label that is nearest to the opening (closing) square bracket corresponds to the label that is nearest to the corresponding closing (opening) square bracket. Each additional label of the square bracket corresponds to a label of an angle bracket (Figure 4).

We will now give axioms that check that the labels of square brackets corresponds to the labels of the matching square and angle brackets:

Axiom 13. (a)+(b) The number of left (right) angle brackets matching each right (left) square bracket is determined by the number of labels associated with the right (left) square bracket.

Axiom 14. (a) Every label of the *right square brackets* has a corresponding bracket with a corresponding label. (b) Every label of the *left square brackets* has a corresponding bracket with a corresponding label.

These axiom are formulated as follows:

$$(\Sigma - L)LL^i\,]_p \Rightarrow [_p\Delta'_{d,p}(\ \rangle_p\Delta'_{d,p})^i\ \_\_ \quad (13a)$$

$$[_pLL^i(\Sigma - L) \Rightarrow \ \_\_\Delta''_{d,p}(\ \rangle_p\Delta''_{d,p})^i\,]_p. \quad (13b)$$

where $0 \leq i < r, 1 \leq p \leq c$,

$$Q_{p,s,i} = \begin{cases} (\Delta_{0,p} \cup [_p\ q_{p,s}\ ]_p)^* \cap \Delta_{d,p} & \text{if } i = d; \\ (\Delta_{0,p} \cup [_p\ Q_{p,s,i+1}\ ]_p)^* \cap \Delta_{d,p} & \text{if } i < d; \end{cases} \quad (14a)$$

$$R_{p,s,i} = \begin{cases} (\Delta_{0,p} \cup [_p\ r_{p,s}\ ]_p)^* \cap \Delta_{d,p} & \text{if } i = d; \\ (\Delta_{0,p} \cup [_p\ R_{p,s,i+1}\ ]_p)^* \cap \Delta_{d,p} & \text{if } i < d; \end{cases} \quad (14b)$$

where $1 \leq p \leq c, 0 \leq s < r, 1 \leq i \leq d$,

and $q_{p,s}$ and $r_{p,s}$ describe what is inside the $[_p,]_p$-square brackets, when the $(s+1)^{\text{th}}$ label of the left and right square bracket, respectively, has a matching label. These languages are defined as

$$q_{p,s} = L^sT_p(\ \rangle_p\Delta''_{d,p})^s \cup (L^* - L^sL^*)(\Sigma - L)\Delta_{d,p}$$
$$r_{p,s} = (\Delta'_{d,p}\ \langle_p)^sT_pL^s \cup \Delta_{d,p}(\Sigma - L)(L^* - L^sL^*)$$

where

$$T_p = \cup_{x \in L_D}(x\ \Delta_{d,p}\ \overline{x}\ \ \cup\ \ \overline{x}\ \Delta_{d,p}\ x)\ \cap\ \Delta_{d,p}$$

is a language whose strings contain just matching pairs of labels and everything that can come between them.

6 Non-Projectivity Depth

The arcs in dependency trees constitute, by the definition of trees, an acyclic graph — our discussion assumes that there are no secondary links in D-trees. In the axiomatization of the string representation, we have to enforce acyclicity by some constraints. Procedurally the acyclicity could be decided, for example, by trying to arrange the nodes into an order where the arcs go from the left to the right (topological sorting). Corresponding declarative solutions would be *e.g.* (i) to use set constraints (Duchier, 1999) or (ii) to attach each node an integer that increases strictly in the nodes reached by the outgoing arcs of the node. Both of these solutions are problematic because the number of reached nodes is, in practice, unbounded. An alternative solution that is adopted her is to use a monotonically increasing counter that is incremented only at certain critical positions. For technical reasons, we attach such a counter to arcs and brackets rather than to the nodes — this change is not mathematically significant.

Let $\prec$ be the linear precedence relation over the nodes. A node in a D-tree is an *articulation node* if no arcs are passing it and the arcs coming into it are on the opposite side than the arcs going out from it. A chain of colored arcs $v_1 \rightarrowtail_{c_1} v_2, v_2 \rightarrowtail_{c_2} v_3, v_3 \rightarrowtail_{c_3} v_4, \ldots v_{n-1} \rightarrowtail_{c_{n-1}} v_n$, where c_i is the color of an arc $v_i \rightarrowtail_{c_i} v_{i+1}$, is called a *colored dependency path*. A node v_i, $1 < i < n$, is *critical* if either (i) $v_{i+1} \prec v_{i-1} \prec v_i$, (ii) $v_i \prec v_{i-1} \prec v_{i+1}$, or (iii) $c_{i-1} > c_{i+1}$. The *non-projectivity depth* of a (colored dependency) path that does not contain an articulation node is the number of critical nodes visited by it. The maximum depth of such paths in a D-tree is the *non-projectivity depth of the D-tree*.

Incrementing counters only at the critical positions has an important advantage over the other solutions mentioned: projective trees do not contain any critical positions, and in non-projective trees of

natural language sentences we probably need only very small numbers. If the counter is incremented several times, the path can be very "unnatural" as shown in Figure 5. We conjecture that if a dependency path is acyclic, we cannot increment the counters at every critical position. In other words, assigning depths to the arcs of a D-graph excludes the alternative that the graph would be cyclic.

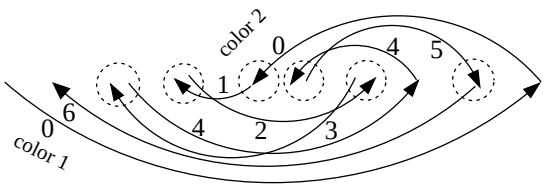

Figure 5: The growth of the non-projectivity depth.

In our representation, there is a fixed upper bound t for the *non-projectivity depth*. Based on it, we can define, for all $i = [0, 1, 2, \ldots, t]$, the sets $L_{D,i} = \{\ (x, i) \mid x \in \Lambda\} \subseteq L_D$, and $L_{H,i} = \{\ \overline{x} \mid x \in L_{D,i}\}$. The second component of each label (x, i) is the non-projectivity depth counter. The incrementation of these counters in critical nodes gives rise to the following axioms. They constrain nodes *i.e. substrings* occurring between two word boundaries:

Axiom 15. (a) In nodes that are not articulation nodes, there is no label $x \in L_{D,i}$, where $0 \leq i < t$, if there is a label $y \in L_{H,j}$, $i < j \leq t$. (b) There is no label $x \in L_{D,j}$, where $0 < j \leq t$, if there is no label $y \in L_H$.

Axiom 16. (a) There are no labels $x \in L_{H,k}$ and $y \in L_{D,k}$, where $0 \leq k \leq t$, that are attached to *closing brackets* in this order. (b) There are no labels $x \in L_{D,k}$ and $y \in L_{H,k}$, where $0 \leq k \leq t$, that are attached to *opening brackets* in this order. (c) There are no labels $x \in L_{D,k}$ and $y \in L_{H,k}$, where $0 \leq k \leq t$, that are attached respectively to a *closing bracket* and an *opening bracket* so that the color index of the closing bracket is smaller than that of the opening bracket.

Axiom 17. There is no label $x \in L_{D,j}$, $0 < j \leq t$, that is attached to an *opening bracket*, if on the left of the label x there is no $y \in L_{H,j}$, or on the right of the label x there is no label $y \in L_{H,j-1}$.

Axiom 18. There is no label $x \in L_{D,j}$, $0 < j \leq t$, that is attached to a *closing bracket with color* p, if on the right of the label x there no label $y \in L_{H,j}$, or on the left of the label x there is no label $y \in L_{H,j-1}$, or there is no label $y \in L_{H,j-1}$ that is attached to an opening bracket with a color greater than p.

Axiom 19. In articulation nodes, the counters of the outgoing arc labels must be zero.

More formally these are given as follows:

$$\Sigma^* - \Sigma^*(E \boxdot (L_{D,i} \boxdot L_{H,j} \cup L_{H,j} \boxdot L_{D,i}) \cup (L_{D,i} \boxdot L_{H,j} \cup L_{H,j} \boxdot L_{D,i}) \boxdot E \cup L_{H,j} \boxdot E' \boxdot L_{D,i} \cup L_{D,i} \boxdot E' \boxdot L_{H,j})\Sigma^* \quad (15a)$$

$$L_{D,j} \Rightarrow L_H \boxdot \_\_, \_\_ \boxdot L_H \quad (15b)$$

$$\Sigma^* - \Sigma^* L_{H,k} \boxdot L_{D,k} \boxdot (B_R \cup A_R)\Sigma^* \quad (16a)$$

$$\Sigma^* - \Sigma^*(B_L \cup A_L) \boxdot L_{D,k} \boxdot L_{H,k}\Sigma^* \quad (16b)$$

$$\Sigma^* - \Sigma^* L_{D,k} L^* B' \boxdot \{[_q, \langle_q\} L^* L_{D,k} \Sigma^* \quad (16c)$$

$$L_{D,j} \Rightarrow L_{H,j} \boxdot \_\_, \quad \_\_(\boxdot \cup L_{H,j-1}) \cup L^*(B_R \cup A_R) \quad (17)$$

$$L_{D,j} \Rightarrow L_{H,j-1} \boxdot \_\_, \quad \_\_(\boxdot(L_{H,j} \cup B' L^* L_{H,j-1})) \cup (\Sigma^{*\#} - L^*\{]_p, \rangle_p\}\Sigma^*) \quad (18)$$

$$L_{D,j} \Rightarrow E \boxdot \_\_ \boxdot L_H, L_H \boxdot \_\_ \boxdot E, \quad (E \boxdot L_H \cup E') \boxdot \_\_, \_\_ \boxdot (L_H \boxdot E \cup E') \quad (19)$$

where $0 \leq i < j \leq t, 0 \leq k \leq t, 1 \leq p < c$, and $B' = \{[_q, \langle_q \mid p < q \leq c\}$, $E' = E - \{1{:}\}$

7 Colored Non-projective Dependency Grammar

A new grammatical framework, called *Colored Non-projective Dependency Grammar (CNDG)* (Yli-Jyrä and Nykänen, 2004), has been developed on the top of the bounded nested crossing depth and bounded non-projectivity depth using a carefully designed linear context free rewriting system. A regular approximation for such a grammar is obtained by compiling the colored dependency rules to constraints that specify local subcategorization features (labels and bracket colors) within the node boundaries. The current axioms will take care of the non-local structure of the described D-trees.

Example 1. The following set of rules describes the dependency tree shown in Figure 1:

$$*(\ 0/\text{pred})$$
$$()\ *$$
$$\text{venit}(\ 0/\overline{\text{pred}}\ \ *\ \ 2/\text{adv}\ \ \text{subj}\ )$$
$$\text{aetas}(\ 1/\text{attr}\ \ 2/\overline{\text{subj}}\ \ \text{gntv}\ \ *\ )$$
$$\text{iam}(\ 2/\overline{\text{adv}}\ \ *\ )$$
$$\text{ultima}(\ *\ \ 1/\overline{\text{attr}}\ )$$
$$\text{Cumaei}(\ *\ \ 1/\overline{\text{attr}}\ )$$
$$\text{carminis}(\ 1/\text{attr}\ \ *\ \ 2/\overline{\text{gntv}}\ )$$

When the second and the third rule, for example, are compiled, we obtain the following two constraints:

$$\overline{\text{pred}} \Rightarrow {}^{\#}\ \_\_\ ]_0(\text{venit} \cup E)^*\ \{[_2, \langle_2\}\ \text{subj}\ [_2\ {}^*\ \text{adv}\ {}^{\#}$$
$$\overline{\text{subj}} \Rightarrow {}^{\#}\ \text{gntv}\ ]_2\ {}^*\ \_\_\{]_2, \rangle_2\}\ \text{attr}\ ]_1\ (\text{aetas} \cup E)^*\ {}^{\#}.$$

8 Further Work

In the future, the current axiomatization should be extended to allow free dependents, and to include rules without colors and arc order. Furthermore, efficient methods for applying the axioms should be developed and a standard finite-state parser using these axioms should be specified.

The approach could be extended with a multi-tiered approach where different kinds of bracketed strings (including *e.g.* P-markers) are processed with a multi-tape finite automaton. We could also use weighted automata to improve the ranking of alternative analyses.

We would like to develop full scale grammars and to evaluate the presented representation properly in practical setting. Possibilities to induce a grammar automatically from a treebank could be examined. The proposed complexity bounds could be applied also to treebank validation and more generally in linguistic studies of natural language complexity.

9 Conclusion

In this article, we have proposed a new representation for restricted non-projective dependency trees: The representation combines both dependency and linearization into a single string structure, and it gives a realistic basis for non-projective finite-state dependency parsers. The complexity measures used here may be of interest in different areas of linguistics.

10 Acknowledgements

This work was funded by NorFA under the author's personal Ph.D. scholarship (ref.nr. 010529). The author is grateful to the anonymous referees for insightful comments on an earlier version of this paper.

References

Steven Abney. 1996. Partial parsing via finite state cascades. In *Proceedings of the ESSLLI'96 Robust Parsing Workshop*.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton, The Hague.

Michael A. Covington. 1990. Technical correspondence. *Computational Linguistics*, 16(4).

Denys Duchier. 1999. Set constraints in computational linguistics – solving tree descriptions. In *Workshop on Declarative Programming with Sets (DPS'99)*, pages 91–98, Paris.

David Elworthy. 2000. A finite state parser with dependency structure output. In *Proceedings of International Workshop on Parsing Technologies*.

Sylvain Kahane, Alexis Nasr, and Owen Rambow. 1998. Pseudo-projectivity: A polynomially parsable non-projective dependency grammar. In *COLING-ACL'98*, volume I, pages 646–652.

Fred Karlsson. (in print). Limits of clausal embedding complexity in Standard Average European. Manuscript. Department of General Linguistics, University of Helsinki, April 2004.

Kimmo Koskenniemi. 1997. Representations and finite-state components in natural language. In E. Roche and Y. Schabes, editors, *Finite-state language processing*, pages 99–116. A Bradford Book, MIT Press, Cambridge, MA.

Steven Krauwer and Louis des Tombe. 1981. Transducers and grammars as theories of language. *Theoretical Linguistics*, 8:173–202.

Igor A. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press, Albany.

Alexis Nasr, Owen Rambow, John Chen, and Srinivas Bangalore. 2002. Context-free parsing of a tree adjoining grammar using finite-state machines. In *Proceedings of TAG+6*, pages 100–105, Universitá di Venezia.

Kemal Oflazer. 2003. Dependency parsing with an extended finite-state approach. *Computational Linguistics*, 29(4).

Anssi Yli-Jyrä and Matti Nykänen. 2004. A hierarchy of mildly context sensitive dependency grammars. In *Formal Grammar (FGNancy)*, Nancy, France, August.

Anssi Yli-Jyrä. 2003a. Multiplanarity - a model for dependency structures in treebanks. In *The Second Workshop on Treebanks and Linguistic Theories*, Växjö, Sweden, 14-15 November.

Anssi Yli-Jyrä. 2003b. Regular approximations through labeled bracketing. In Gerhard Jäger, Paola Monachesi, Gerald Penn, and Shuly Wintner, editors, *Proceedings of Formal Grammar 2003*, pages 189–201, Vienna.

Anssi Yli-Jyrä. 2004a. Approximating dependency grammars through intersection of regular string languages. Ninth International Conference on Implementation and Application of Automata (CIAA), July.

Anssi Yli-Jyrä. 2004b. Coping with dependencies and word order or how to put Arthur's court into a castle. In H. Holmboe, editor, *Nordisk Sprogteknologi 2003. Årbog for Nordisk Sprogteknologisk Forskningsprogram 2000-2004*, pages 123–137. Museum Tusculanums Forlag, Københavns Universitet.