Relationship between Non-Projective Edges, their Level Types, and Well-Nestedness

Jiří Havelka
Institute of Formal and Applied Linguistics
Charles University in Prague, Czech Republic
havelka@ufal.mff.cuni.cz

Abstract

Dependency analysis of natural language gives rise to non-projective structures. The constraint of well-nestedness on dependency trees has been recently shown to give a good fit with empirical linguistic data. We present a reformulation of this constraint using properties of non-projective edges and show its formal relationship to level types of non-projective edges; we also derive a simple $O(n^2)$ algorithm for checking well-nestedness.

1 Introduction

Dependency analysis of natural language has been gaining an ever increasing interest thanks to its applicability in many tasks of NLP—a recent example is the dependency parsing work of McDonald et al. (2005), which introduces an approach based on the search for maximum spanning trees, capable of handling non-projective structures naturally.

In this context, the issue of delimiting dependency structures permissible in natural language grows in importance (see e.g. Nivre (2006)). We would like to point out that since neither syntactic structures in dependency treebanks, nor structures arising in dependency parsing need a priori fall into any formal subclass of dependency trees, we need means of describing any non-projective structure.[1]

Kuhlmann and Nivre (2006) compare several constraints on dependency structures and among the considered ones find well-nestedness to be in good accord with empirical data. However, they do not include level types of non-projective edges introduced by Havelka (2005), which present another means of characterizing any non-projective structure and have interesting formal properties. We link properties of non-projective edges and their level types to the constraint of well-nestedness and show that they provide a more fine-grained means capable of capturing it.

The paper is organized as follows: Sect. 2 contains formal preliminaries; Sect. 3 and 4 review definitions and show the necessary properties of the constraint of well-nestedness and level types of non-projective edges; Sect. 5 presents the main results concerning the relationship between non-projective edges (and their level types) and well-nestedness.

2 Formal preliminaries

To make the paper as self-contained as possible, we provide a concise reference with definitions and simple properties used in subsequent sections.

Definition 1 A *dependency tree* is a triple $(V, \to, \preceq)$, where V is a finite set of nodes, $\to$ a *dependency* relation on V, and $\preceq$ a total order on V.

Relation $\to$ models linguistic dependency, and so represents a directed, rooted tree on V. There are many ways of characterizing rooted trees, we give here a characterization via the properties of $\to$: there is a *root* $r \in V$ such that $r \to^* v$ for all $v \in V$ and there

[1]The importance of such means is evident from the asymptotically negligible proportion of projective trees to all dependency trees. (Unrestricted dep. trees (i.e. labelled rooted trees), well-nested dep. trees, and projective dep. trees are counted by sequences A000169, A113882, and A006013 (offset 1), resp., in the On-Line Encyclopedia of Sequences (Sloane, 2007).)

is a unique *edge* $p \to v$ for all $v \in V$, $v \neq r$. Relation $\to^*$ is the reflexive transitive closure of $\to$ and is usually called *subordination*.

The following definitions allow us to formulate our results succinctly. For each node i we define its *level* as the length of the path $r \to^* i$; we denote it level_i. The symmetrization $\leftrightarrow \,=\, \to \cup \to^{-1}$ makes it possible to talk about edges (pairs of nodes i, j such that $i \to j$) without explicitly specifying the *parent* (*head* etc.; i here) and the *child* (*dependent* etc.; j here); so $\to$ represents directed edges and $\leftrightarrow$ undirected edges. To retain the ability to talk about the direction of edges, we define $\mathsf{Parent}_{i\leftrightarrow j} = \begin{cases} i & \text{if } i \to j \\ j & \text{if } j \to i \end{cases}$ and $\mathsf{Child}_{i\leftrightarrow j} = \begin{cases} j & \text{if } i \to j \\ i & \text{if } j \to i \end{cases}$. Our notation for rooted *subtrees* is $\mathsf{Subtree}_i = \{v \in V \mid i \to^* v\}$, $\mathsf{Subtree}_{i\leftrightarrow j} = \{v \in V \mid \mathsf{Parent}_{i\leftrightarrow j} \to^* v\}$, and for *ancestors* $\mathsf{Anc}_i = \{v \in V \mid v \to^* i\}$, $\mathsf{Anc}_{i\leftrightarrow j} = \{v \in V \mid v \to^* \mathsf{Parent}_{i\leftrightarrow j}\}$. To be able to talk concisely about the total order on nodes $\preceq$, we define *open* and *closed intervals* whose endpoints need not be in a prescribed order: $(i, j) = \{v \in V \mid \min_{\preceq}\{i, j\} \prec v \prec \max_{\preceq}\{i, j\}\}$ and $[i, j] = \{v \in V \mid \min_{\preceq}\{i, j\} \preceq v \preceq \max_{\preceq}\{i, j\}\}$, resp. For any edge $i \leftrightarrow j$ we define its *gap* as follows $\mathsf{Gap}_{i\leftrightarrow j} = \{v \in V \mid v \in (i, j) \;\&\; v \notin \mathsf{Subtree}_{i\leftrightarrow j}\}$. An edge with an empty gap is *projective*, an edge whose gap is non-empty is *non-projective*. (See e.g. (Havelka, 2005) for the characterization of projectivity via properties of edges and further references.)

Property 2 *Let a be a node and $i \leftrightarrow j$ any edge disjoint from a. Then $i \in \mathsf{Subtree}_a \Leftrightarrow j \in \mathsf{Subtree}_a$.*
PROOF. From the assumption $i \neq a \neq j$ it follows that $i, j \in \mathsf{Subtree}_a \Leftrightarrow \mathsf{Parent}_{i\leftrightarrow j} \in \mathsf{Subtree}_a$. ∎

Proposition 3 *Let $i \leftrightarrow j$, $u \leftrightarrow v$ be disjoint edges.*
(i) *If $u, v \in (i, j)$, then $u \in \mathsf{Gap}_{i\leftrightarrow j} \Leftrightarrow v \in \mathsf{Gap}_{i\leftrightarrow j}$.*
(ii) *If $u \in \mathsf{Gap}_{i\leftrightarrow j}$ and $v \notin \mathsf{Gap}_{i\leftrightarrow j}$, then $v \notin [i, j]$.*
PROOF. (i) follows immediately from the definition of $\mathsf{Gap}_{i\leftrightarrow j}$ and Property 2. To prove (ii), assume $v \in (i, j)$ and using (i) arrive at a contradiction. ∎

3 Well-nestedness

Kuhlmann and Nivre (2006) claim that the constraint of well-nestedness seems to approximate well dependency structures occurring in natural language.

Definition 4 A dependency tree T is *ill-nested* if there are disjoint subtrees T_1, T_2 of T and nodes $x_1, y_1 \in T_1$ and $x_2, y_2 \in T_2$ such that $x_1 \in (x_2, y_2)$ and $x_2 \in (x_1, y_1)$. A dependency tree T that is not ill-nested is *well-nested*.[2]

It is easy to express the constraint in terms of edges—it will prove crucial in Sect. 5.

Theorem 5 *A dependency tree T is ill-nested iff there are edges $i_1 \leftrightarrow j_1$, $i_2 \leftrightarrow j_2$ in disjoint subtrees T_1, T_2 of T, resp., such that $i_1 \in (i_2, j_2)$, $i_2 \in (i_1, j_1)$.*
PROOF. Direction $\Leftarrow$ is obvious.

Direction $\Rightarrow$: Let r_i be the root of T_i. To find $i_1 \leftrightarrow j_1$, first suppose that $r_1 \in (x_2, y_2)$. Consider the first edge $v_k \to v_{k+1}$ on the downward path $v_0 = r_1, v_1, \ldots, v_m = y_1$, $m > 0$, such that $v_k \in (x_2, y_2)$ and $v_{k+1} \notin [x_2, y_2]$. If $r_1 \notin [x_2, y_2]$, consider the first edge $v_{k+1} \to v_k$ on the upward path $v_0 = x_1, v_1, \ldots, v_n = r_1$, $n > 0$, such that $v_k \in (x_2, y_2)$ and $v_{k+1} \notin [x_2, y_2]$. Let us denote $i_1 = v_k$ and $j_1 = v_{k+1}$, and possibly rename x_2, y_2 so that $i_1 \in (x_2, y_2)$ and $x_2 \in (i_1, j_1)$. To find $i_2 \leftrightarrow j_2$ such that $i_1 \in (i_2, j_2)$, $i_2 \in (i_1, j_1)$, proceed similarly as above. Obviously, edges $i_1 \leftrightarrow j_1$, $i_2 \leftrightarrow j_2$ are in disjoint subtrees. ∎

4 Level types of non-projective edges

Level types of non-projective edges allow their structural classification with interesting formal properties. They were introduced by Havelka (2005), who presents them in more detail.

Definition 6 The *level type* (or just *type*) of a non-projective edge $i \leftrightarrow j$ is defined as follows

$$\mathsf{Type}_{i\leftrightarrow j} = \mathsf{level}_{\mathsf{Child}_{i\leftrightarrow j}} - \min_{n \in \mathsf{Gap}_{i\leftrightarrow j}} \mathsf{level}_n \,.$$

The type of an edge is the distance of its child node and a node in its gap closest to the root (distance here means difference in levels)—for sample configurations see Figure 1[3]. Note that there may be more than one node witnessing an edge's type. The type of an edge is not bounded—it can take any integer value (depending on the height of a tree).

Our definition of level type of non-projective edges extends naturally the original definition im-

[2]The original definition requires $x_1 \prec x_2 \prec y_1 \prec y_2$, however our equivalent formulation leads to shorter theorems and proofs.

[3]We adopt the following convention: nodes are drawn top-down according to their increasing level, with nodes on the same level on the same horizontal line; nodes are drawn from left to right according to the total order on nodes; edges are drawn as solid lines, paths as dotted curves. We assume that no node on a path lies in the span of an edge the path crosses.

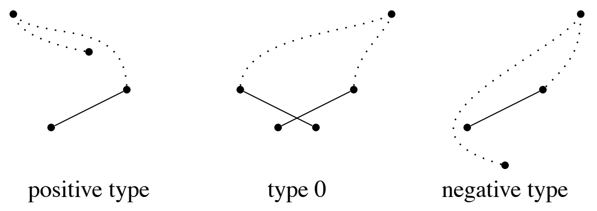
Figure 1: Sample non-projective edges

plicit in (Havelka, 2005), where classes corresponding to positive, zero and negative types are defined.

We now show a relationship between edges of certain types that will allow us to derive a relationship between non-projective edges and well-nestedness.

Theorem 7 *For any non-projective edge $i \leftrightarrow j$ in a dependency tree T with $\mathsf{Type}_{i\leftrightarrow j} \leq 0$ (< 0) there is a non-projective edge $v \rightarrow u$ in T with $\mathsf{Type}_{u\leftrightarrow v} \geq 0$ (> 0) such that $u \in \arg\min_{n\in\mathsf{Gap}_{i\leftrightarrow j}} \mathsf{level}_n$ and either $i \in \mathsf{Gap}_{u\leftrightarrow v}$, or $j \in \mathsf{Gap}_{u\leftrightarrow v}$.*

PROOF. Let u be any node in $\arg\min_{n\in\mathsf{Gap}_{i\leftrightarrow j}} \mathsf{level}_n$. From the assumption $\mathsf{Type}_{i\leftrightarrow j} \leq 0$ node u has a parent $v \notin \mathsf{Gap}_{i\leftrightarrow j}$. Obviously $i \leftrightarrow j$, $v \rightarrow u$ are disjoint, thus from Proposition 3 we have $v \notin [i,j]$, and so either $i \in (u,v)$, or $j \in (u,v)$. Since $\mathsf{level}_v \geq \mathsf{level}_{\mathsf{Parent}_{i\leftrightarrow j}}$, we have that $\mathsf{Parent}_{i\leftrightarrow j} \notin \mathsf{Subtree}_v$, and so either $i \in \mathsf{Gap}_{u\leftrightarrow v}$, or $j \in \mathsf{Gap}_{u\leftrightarrow v}$. Finally from $\mathsf{Type}_{i\leftrightarrow j} \leq 0$ (< 0) we get $\mathsf{level}_u - \mathsf{level}_{\mathsf{Child}_{i\leftrightarrow j}} \geq 0$ (> 0), hence $\mathsf{Type}_{u\leftrightarrow v} \geq 0$ (> 0). ∎

5 Well-nestedness & non-projective edges

We give characterizations of well-nestedness solely in terms of properties of non-projective edges and show some applications.

5.1 Characterization using pairs of edges

First we give a characterization of pairs of edges in Theorem 5 in terms of their gaps.

Theorem 8 *Let $i_1 \leftrightarrow j_1$, $i_2 \leftrightarrow j_2$ be two edges in a dependency tree T. They are in disjoint subtrees T_1, T_2, resp., and satisfy $i_1 \in (i_2, j_2)$, $i_2 \in (i_1, j_1)$ iff the following condition holds*

$$(\mathsf{inp}) \qquad i_1 \in \mathsf{Gap}_{i_2\leftrightarrow j_2} \ \ \& \ \ i_2 \in \mathsf{Gap}_{i_1\leftrightarrow j_1} \ .$$

PROOF. Direction $\Leftarrow$: Root T_k in $\mathsf{Parent}_{i_k\leftrightarrow j_k}$, $k = 1,2$. Condition (inp) obviously implies $i_1 \in (i_2, j_2)$, $i_2 \in (i_1, j_1)$, which in turn implies that edges $i_1 \leftrightarrow j_1$, $i_2 \leftrightarrow j_2$ are disjoint. From Property 2 we get that both $\mathsf{Parent}_{i_2\leftrightarrow j_2} \notin \mathsf{Subtree}_{i_1\leftrightarrow j_1}$ and $\mathsf{Parent}_{i_1\leftrightarrow j_1} \notin \mathsf{Subtree}_{i_2\leftrightarrow j_2}$, hence subtrees T_1, T_2 are disjoint.

Direction $\Rightarrow$: Let us consider the edge $i_2 \leftrightarrow j_2$ and node i_1. Since T_1 is disjoint from T_2, we have that $i_1 \notin \mathsf{Subtree}_{i_2\leftrightarrow j_2}$, and therefore $i_1 \in \mathsf{Gap}_{i_2\leftrightarrow j_2}$. The proof that $i_2 \in \mathsf{Gap}_{i_1\leftrightarrow j_1}$ is analogous. ∎

Condition (inp) allows us to talk about pairs of edges causing ill-nestedness and so characterize well-nestedness using properties of pairs of edges.

Definition 9 We say that any two non-projective edges $i_1 \leftrightarrow j_1$, $i_2 \leftrightarrow j_2$ in a dependency tree T satisfying condition (inp) form an *ill-nested pair of edges*.

Corollary 10 *A dependency tree T is ill-nested iff it contains an ill-nested pair of edges.*

PROOF. Follows from Theorems 5 and 8. ∎

5.2 Sufficient condition for ill-nestedness

The results of Section 4 and previous subsection give the following relationship between types of non-projective edges and well-nestedness.

Theorem 11 *If a dependency tree contains a non-proj. edge of nonpositive type, then it is ill-nested.*

PROOF. Follows from Theorems 7 and 10. ∎

We see that types of non-projective edges and well-nestedness share a common ground; however, the statement of Theorem 11 cannot be strengthened to equivalence (it is easy to see that also two edges of positive type can satisfy (inp)).

5.3 Characterization using single edges

Now we show that well-nestedness can be characterized in terms of properties of single non-projective edges only. We define the ill-nested set of an edge and show that it gives the desired characterization.

Definition 12 The *ill-nested set* of any edge $i \leftrightarrow j$ is defined as follows

$$\mathsf{In}_{i\leftrightarrow j} = \{u \leftrightarrow v \mid u \in \mathsf{Gap}_{i\leftrightarrow j} \ \& \ v \notin [i,j] \ \& \ u,v \notin \mathsf{Anc}_{i\leftrightarrow j}\} \ .$$

The next proposition exposes the relationship of edges in $\mathsf{In}_{i\leftrightarrow j}$ to the gap of $i \leftrightarrow j$.

Proposition 13 *For any edge $i \leftrightarrow j$ $\mathsf{In}_{i\leftrightarrow j} = \{u \leftrightarrow v \mid u \in \mathsf{Gap}_{i\leftrightarrow j} \ \& \ v \notin \mathsf{Gap}_{i\leftrightarrow j} \ \& \ u,v \notin \mathsf{Anc}_{i\leftrightarrow j}\}$.*

PROOF. The inclusion $\subseteq$ is obvious. The inclusion $\supseteq$ follows from Proposition 3 ($u \in \mathsf{Gap}_{i\leftrightarrow j}$ and $v \notin \mathsf{Anc}_{i\leftrightarrow j}$ imply that edges $i \leftrightarrow j$, $u \leftrightarrow v$ are disjoint). ∎

We are ready to formulate the main result of this section, which gives as corollary a characterization of well-nestedness using properties of single edges.

Theorem 14 *Let $i \leftrightarrow j$ be an edge in a dependency tree T. The edges that form an ill-nested pair with the edge $i \leftrightarrow j$ are exactly the edges in $\mathsf{In}_{i\leftrightarrow j}$.*

PROOF. Direction $\Rightarrow$: Let $u \leftrightarrow v$ be an edge forming an ill-nested pair with the edge $i \leftrightarrow j$, i.e. $i \in \mathsf{Gap}_{u\leftrightarrow v}$ and $u \in \mathsf{Gap}_{i\leftrightarrow j}$. This implies $i \in (u,v)$ and $u \in (i,j)$, which immediately gives $v \notin [i,j]$. Supposing $u \in \mathsf{Anc}_{i\leftrightarrow j}$ or $v \in \mathsf{Anc}_{i\leftrightarrow j}$ we get $i \in \mathsf{Subtree}_{u\leftrightarrow v}$, which is in contradiction with $i \in \mathsf{Gap}_{u\leftrightarrow v}$, and therefore $u,v \notin \mathsf{Anc}_{i\leftrightarrow j}$. Hence $u \leftrightarrow v \in \mathsf{In}_{i\leftrightarrow j}$.

Direction $\Leftarrow$: Let $u \leftrightarrow v \in \mathsf{In}_{i\leftrightarrow j}$ (i.e. $u \in \mathsf{Gap}_{i\leftrightarrow j}$, $v \notin [i,j]$, and $u,v \notin \mathsf{Anc}_{i\leftrightarrow j}$; without loss of generality assume $i \in (u,v)$). From the assumptions $u \in \mathsf{Gap}_{i\leftrightarrow j}$ and $v \notin [i,j]$ we get that edges $i \leftrightarrow j$, $u \leftrightarrow v$ are disjoint. Using Property 2, from the assumption $u,v \notin \mathsf{Anc}_{i\leftrightarrow j}$ we get $i \notin \mathsf{Subtree}_{u\leftrightarrow v}$, thus $i \in \mathsf{Gap}_{u\leftrightarrow v}$. Hence $i \leftrightarrow j$, $u \leftrightarrow v$ satisfy (inp). ■

Corollary 15 *A dependency tree T is ill-nested iff $\mathsf{In}_{i\leftrightarrow j} \neq \emptyset$ for some non-projective edge $i \leftrightarrow j$ in T.*

PROOF. Follows from Theorems 8 and 14. ■

5.4 Checking well-nestedness

Our characterization of well-nestedness gives also a novel way of checking it. Here is a pseudocode of an algorithm for fully determining all ill-nested sets:

```
1: for all edges $i \leftrightarrow j$ do
2:     for all edges $u \leftrightarrow v$ s.t. $u \in (i,j)$ do
3:         check $u \leftrightarrow v \in \mathsf{In}_{i\leftrightarrow j}$
```

Its time complexity is obviously $O(n^2)$, since the check on line 3 can be implemented so as to take constant time (by precompuing $\rightarrow^*$, which can be done in $O(n^2)$ time). The bound is the same as for the reported algorithms for checking well-nestedness (Möhl, 2006).

However, the following theorem allows well-nestedness checking to be linear for projective trees, to be faster for random input, and to remain $O(n^2)$.

Theorem 16 *In any ill-nested pair of edges, at least one of the edges is of nonnegative type (witnessed by an end-point of the other edge).*

PROOF. Let $i_1 \leftrightarrow j_1$, $i_2 \leftrightarrow j_2$ satisfy (inp). Let us suppose that $\mathsf{level}_{\mathsf{Child}_{i_1\leftrightarrow j_1}} \geq \mathsf{level}_{\mathsf{Child}_{i_2\leftrightarrow j_2}}$. Since $\mathsf{level}_{\mathsf{Child}_{u\leftrightarrow v}} \geq \mathsf{level}_u$ for any edge $u \leftrightarrow v$, we have that $\mathsf{level}_{\mathsf{Child}_{i_1\leftrightarrow j_1}} \geq \mathsf{level}_{i_2}$, and hence $\mathsf{Type}_{i_1\leftrightarrow j_1} \geq 0$. If $\mathsf{level}_{\mathsf{Child}_{i_1\leftrightarrow j_1}} \leq \mathsf{level}_{\mathsf{Child}_{i_2\leftrightarrow j_2}}$, it is analogously proved that $i_2 \leftrightarrow j_2$ is of nonnegative type. ■

Havelka (2005) presents a linear algorithm for finding all non-projective edges of nonnegative type. Thus well-nestedness can be checked as follows: first find all edges of nonnegative type, and then check their ill-nested sets for non-emptiness. Computing $\rightarrow^*$ on demand for subtrees of the processed edges, we preserve worst-case quadratic complexity.

6 Conclusion

We have presented new formal results linking properties of non-projective edges and their level types to well-nestedness. This work extends the current body of research on non-projective dependency structures in natural language. In particular, we offer new insights into formal properties of non-projective edges that, if possible, both provide adequate means for linguistic description and at the same time are useful as features in machine-learning approaches.

Acknowledgement This work was supported by projects 1ET201120505 of the Ministry of Education of the Czech Republic and 374/2005/A-INF/MFF of Grant Agency of Charles University.

References

Jiří Havelka. 2005. Projectivity in Totally Ordered Rooted Trees: An Alternative Definition of Projectivity and Optimal Algorithms for Detecting Non-Projective Edges and Projectivizing Totally Ordered Rooted Trees. *Prague Bulletin of Mathematical Linguistics*, 84:13–30.

Marco Kuhlmann and Joakim Nivre. 2006. Mildly Non-Projective Dependency Structures. In *Proceedings of COLING/ACL*, pages 507–514.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of HLT/EMNLP*, pages 523–530.

Mathias Möhl. 2006. Drawings as models of syntactic structure: Theory and algorithms. Diploma thesis, Programming Systems Lab, Universität des Saarlandes, Saarbrücken.

Joakim Nivre. 2006. Constraints on Non-Projective Dependency Parsing. In *Proc. of EACL*, pages 73–80.

Neil J. A. Sloane. 2007. On-Line Encyclopedia of Integer Sequences. Published electronically at `www.research.att.com/~njas/sequences/`.