D-trivial Dependency Grammars with Global Word-Order Restrictions *

Radu Gramatovici[1], Martin Plátek[2]

[1] Faculty of Mathematics, University of Bucharest
14, Academiei st., RO-70109, Bucharest, Romania
Email: radu@funinf.cs.unibuc.ro

[2] Department of Theoretical Computer Science and Mathematical Logic, Faculty of Mathematics and Physics, Charles University
25, Malostránske nam., CZ-11800, Prague, Czech Republic
Email: platek@ksi.ms.mff.cuni.cz

Abstract

An infinite sequence of incomparable classes of semilinear languages bounded between the class of regular and respectively context-sensitive languages is introduced. This sequence is obtained by the relaxation of word order in a sentence represented by a (possibly) non-projective DR-tree, while keeping the projectivity of the corresponding D-tree. Our results argue a significant difference between the (non)projectivity constraints of DR-trees and respectively D-trees and focus on the intrinsic importance of DR-trees.

*This work is supported by the grant of GAČR No. 201/02/1456, by the project of the Ministry of Education of the Czech Republic No. LN00A063 and in part by the grant of GA of Charles University, Grant-No. 300/2002/A INF/MFF.

1 Introduction

The notion of free-order dependency grammar (or simply *dependency grammar*) was introduced in [3] as a formal system suitable for a dependency-based parsing of natural languages. In a certain way this notion enriches the types of dependency grammars described in [1].

The proposal of this system was based upon the experience acquired during the development of a grammar-checker for Czech ([2]) and as a possible next step towards a complete syntactic analysis following the underlying ideas of the dependency-based framework of Functional Generative Description - FGD ([5]). Compared to FGD and other usual formal systems describing the syntax of natural languages, the framework introduced by dependency grammars takes a serious account for the freedom of word order in a sentence and assign the same importance to linear precedence (LP) rules as to immediate dominance (ID) rules.

As the freedom of word order is not total, even in so-called *free-word-order languages*, one needs to constrain the formalism in order to not over-generate the actual language.

In [4], a measure for the freedom of the word order was studied, based on the number of gaps issued in a sentence by the order of their words (*node-gaps-complexity*). Both *global* and *local constraints* on the maximal number of gaps at some node in the structure underlying the sentence were studied. In the view of node-gaps-complexity, word order relaxation means to stepwisely relax the constraints in order to obtain more complex language constructions. In this paper, we work only with global constraints.

Two types of syntactic structures are used in the relationship with dependency grammars, **DR**-*trees* (**D**elete **R**ewrite trees) and **D**-*trees* (**D**ependency trees). If D-trees concern the dependency structure of the sentence, DR-trees rather concern the generation/parsing of the sentence. The two types of structures are related by the fact that any DR-tree can be transformed in an uniform way into a D-tree. The measure for the number of gaps in a sentence computed in the nodes of the structure is introduced for both DR-trees and D-tree. A DR-tree (D-tree) with no gaps is called *projective*, while a DR-tree (D-tree) with at least one gap is called *non-projective*. In this paper, we work only with dependency grammars which possibly create non-projective DR-trees, but cannot create non-projective D-trees (*D-trivial dependency grammars*).

The main result of this paper presents an infinite sequence of incom-

parable classes of semilinear languages generated by a particular type of dependency grammar, called *D-trivial left dependency grammar* (Section 4). Moreover, all these classes of languages are strictly contained between the class of regular languages and the class of context-sensitive languages, each of them containing context-free and non-context-free languages (Section 5).

We prove by the results we obtain in this paper that there is a significant difference between the projectivity of DR-trees and respectively D-trees. We also try to argue in this way the importance of the non-projective generation/parsing of the sentence, which is represented by the DR-tree compared to the non-projectivity of the sentence itself which is represented by the D-tree. In other word, even if is a sentence is represented by a projective D-tree, this D-tree can hide some non-projective concurrency phenomena raising in the generation or the parsing of the sentence.

2 DR-trees and D-trees

In this section, we introduce some basic definitions and results on DR-trees and D-trees. We start with a general notion of annotated tree which underlies both DR-trees and D-trees notions.

Let M be a set. Let $Tr = (Nod, Ed, Rt, Ann)$ be a 4-tuple , where Nod is a set (*the set of nodes*), $Rt \in Nod$ is a special node (*the root*), $Ed : Nod \setminus \{Rt\} \rightarrow Nod$ is a function (*the set of edges*) and $Ann : Nod \rightarrow M$ is a function (*the annotation function*).

We call **path** in Tr any sequence of nodes from Nod,

$$p = (n_1, n_2, \ldots, n_k),$$

with $k \geq 1$, such that $Ed(n_i) = n_{i+1}$, for $i = 1, \ldots k-1$. We say that p is a *path of length* $k-1$ *from* n_1 *to* n_k. If $k > 1$, we denote p by $Path(n_1, n_k)$.

We say that Tr is a **M-annotated tree** iff there is no path of positive length in Tr from a node $n \in Nod$ to itself.

Let $n \in Nod$ be a node in Tr. We say that n is a **leaf** iff $n \neq Ed(n')$, for any $n' \in Nod$.

We say that Tr is a **finite** M-annotated tree iff its set of nodes, Nod, is finite.

In the sequel, we will work only with finite annotated trees, without clearly mention it.

Proposition 2.1 *Let Tr = (Nod, Ed, Rt, Ann) be an annotated tree. Then for any node $n \in Nod$ there exists exactly one path from n to Rt in Tr.*

Proof Let $n \in Nod$ be a node. Let no be the total number of nodes in Nod.

We first prove that any path originating in n has the length less than no. Suppose to a contradiction that there exists a path $p = (n_1, n_2, \ldots, n_k)$ of length at least equal to no originating in n ($k - 1 \geq no$, $n_1 = n$). Then p contains at least $no + 1$ nodes. Since the total number of nodes in Nod is no, there exist two indices $1 \leq i < j \leq k$ such that $n_i = n_j$. Then there exists a path $p' = (n_i, n_{i+1}, \ldots, n_j)$ of positive length from n_i to itself. This fact leads to a contradiction with the hypothesis that Tr is a annotated tree.

It results that any path originating in n has the length less than no. Consider the longest path $p = (n_1, n_2, \ldots, n_k)$ originating in n. Suppose to a contradiction that $n_k \neq Rt$. In this case we consider $n_{k+1} = Ed(n_k)$. Then $p' = (n_1, n_2, \ldots, n_k, n_{k+1})$ is a longer path than p and still originating in n, which leads to a contradiction.

It follows that $n_k = Rt$, hence there exists a path from n to Rt in Tr. The uniqueness comes from the fact that from each node $m \in Nod$ there is exactly one edge in Tr which leads to another node (this relationship is given by Ed, which is a function).

□

Corollary 2.2 *Let Tr = (Nod,Ed,Rt,Ann) be a M-annotated tree. Then for any two nodes $n_1, n_2 \in Nod$ there exists at least one node $n_3 \in Nod$ such that there exist a path from n_1 to n_3 and a path from n_2 to n_3.*

Proof Obviously, we can take $n_3 = Rt$.

□

Remark 2.1 Keeping the denotations from Corollary 2.2, we denote by $sup(n_1, n_2)$ the first node in Tr which connects n_1 and n_2, i.e. $(n_1, \ldots, Rt)$ and $(n_2, \ldots, Rt)$ are the paths from n_1 respectively n_2 to Rt, then $sup(n_1, n_2)$ is the first node, which belongs to both of these paths. From the definition of an annotated tree, $sup(n_1, n_2)$ is uniquely defined by this property.

Let $Tr_1 = (Nod_1, Ed_1, Rt_1, Ann_1)$ and $Tr_2 = (Nod_2, Ed_2, Rt_2, Ann_2)$ be two M-annotated trees. We say that Tr_1 and Tr_2 are **equivalent** iff there is a bijection $f : Nod_1 \to Nod_2$ such that:

1. $f(Ed_1(n)) = Ed_2(f(n))$, for any node $n \in Nod_1 \setminus \{Rt_1\}$;
2. $f(Rt_1) = Rt_2$;
3. $Ann_1(n) = Ann_2(f(n))$, for any node $n \in Nod_1$.

We call f an **isomorphism** between Tr_1 and Tr_2.

Remark 2.2 *If f is an isomorphism between two M-annotated trees Tr_1 and Tr_2, then f^{-1} is an isomorphism between Tr_2 and Tr_1. Moreover, isomorphism functions establish an equivalence relation on the set of trees annotated by the same set of labels.*

Proposition 2.3 *Let $f : Nod_1 \to Nod_2$ be an isomorphism between two M-annotated trees $Tr_1 = (Nod_1, Ed_1, Rt_1, Ann_1)$, $Tr_2 = (Nod_2, Ed_2, Rt_2, Ann_2)$. Then, Tr_1 and Tr_2 have the same number of leaves and any leaf from Tr_1 is mapped by f in a leaf from Tr_2.*

Proof We suppose to a contradiction that there is a leaf $n \in Nod_1$ such that $f(n) = m$ and m is not a leaf in Nod_2. Then, there is $m' \in Nod_2$ such that $Ed(m') = m$. Since f is a bijection, there is $n' \in Nod_1$ such that $f(n') = m'$. From the fact that $m' \in Nod_2 \setminus \{Rt_2\}$ it results that also $n' \in Nod_1 \setminus \{Rt_1\}$, hence there is $n'' \in Nod_1$ such that $Ed(n') = n''$. Obviously, $n'' \neq n$, because n is a leaf. It follows that

$$f(n'') = f(Ed(n')) = Ed(f(n')) = Ed(m') = m.$$

But also $f(n) = m$, which leads to a contradiction with the fact that f is an injective function.

It results that any leaf from Tr_1 is mapped by f in a leaf from Tr_2. Since f is injective, it follows that the number of leaves in Tr_2 is at least equal to the number of leaves in Tr_1.

The other part of the proof can be done in the same way considering the isomorphism f^{-1} between Tr_2 and Tr_1.

□

In [3], (free-order) dependency grammars were introduced, as a rewriting device over two alphabets, of non-terminals and, respectively, terminals. In its general form, a dependency grammar is able to rewrite both non-terminals and terminals, by a finite set of rules (productions). Through out this paper, we will work with dependency grammars, which rewrite only non-terminals

(in a similar way to the context-free grammars), therefore, we will not use terminals on the lefthand-sides of the rules.

We call **dependency grammar** a structure $G = (N, T, S, P)$ such that N and T are non-empty, finite sets, called the set of *nonterminals* and respectively the set of *terminals*, $S \in N$ is the start symbol and P is a finite set, called the set of *productions* such that

$$P \subseteq (N \times VV \times \{L, R\}) \cup (N \times T),$$

where $V = N \cup T$. Sometimes, we will write:

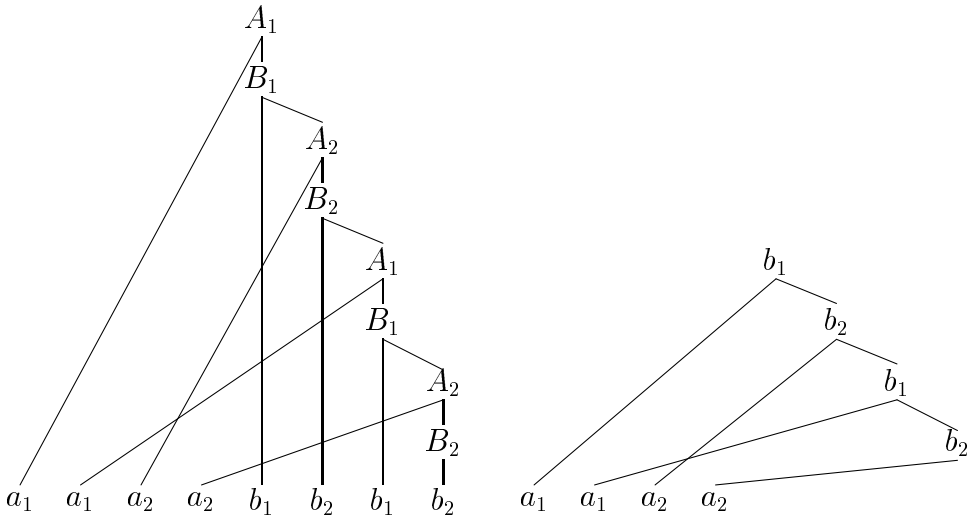
$$\begin{array}{rl} A \to_L BC & \text{instead of } (A, BC, L) \\ A \to_R BC & \text{instead of } (A, BC, R) \\ A \to a & \text{instead of } (A, a) \end{array}$$

for any production of the appropriate form.

Denote by Nat the set of natural numbers not equal to 0 and by $Nat_0 = Nat \cup \{0\}$.

Let $G = (T, N, S, P)$ be a dependency grammar and denote $V = T \cup N$. A **DR-tree created by** G is a V-annotated tree *Tr = (Nod,Ed,Rt,Ann)* such that:

1. $Nod \subseteq Nat \times Nat$.
2. If $Ed(i, j) = (k, l)$ then $j < l$.
3. $Rt = (i, \max\{k \mid \exists j, (j, k) \in Nod\})$.
4. A node $(i, j) \in Nod$ is a leaf if and only if $j = 1$ and $Ann(i, j) \in T$.
5. If $(i, j) \in Nod$, with $j \neq 1$ and $Ann(i, j) = A$, then one of the following cases necessarily occurs:
   a. $j = 2$ and there is exactly one node $n \in Nod$ such that $Ed(n) = (i, j)$; in this case $n = (i, 1)$ and if $Ann(n) = a$, the production $A \to a$ belongs to P.
   b. There are exactly two nodes $n_1, n_2 \in Nod$ such that $Ed(n_1) = Ed(n_2) = (i, j)$; in this case either:
      b1. $n_1 = (i, k)$ and $n_2 = (l, m)$ with $l > i$, $\max(k, m) = j - 1$ and if $Ann(n_1) = B$ and $Ann(n_2) = C$, the production $A \to_L BC$ belongs to P, or



a. DR-tree Tr b. D-tree dTr

Figure 1: Examples of a DR-tree, respectively, a D-tree

b2. $n_1 = (l, k)$ and $n_2 = (i, m)$ with $l < i$, $\max(k, m) = j - 1$ and if $Ann(n_1) = B$ and $Ann(n_2) = C$, the production $A \rightarrow_R BC$ belongs to P.

Let $n_o \in Nod, n_o = (i, j)$. We say that i is the horizontal position of n_o and j is the vertical position of n_o (see an example of a DR-tree in Figure 1).

We say that a DR-tree *Tr = (Nod, Ed, Rt, Ann)* is **complete** iff for any leaf $(i, 1) \in Nod$, if $i > 1$ then also $(i - 1, 1) \in Nod$.

For any complete DR-tree *Tr = (Nod, Ed, Rt, Ann)* created by a dependency grammar $G = (N, T, S, P)$, we define the **sentence** associated with Tr by $s(Tr) = a_1 a_2 \ldots a_n$, where $n = \max\{i \mid (i, 1) \in Nod\}$ and $Ann(i, 1) = a_i$, for any $i \in [n]$. Obviously, $s(Tr) \subseteq T^+$.

Let $G = (N, T, S, P)$ be a dependency grammar. We denote by:

- $T(G)$ the set of complete DR-trees created by G and rooted by S;

- *DR-L*$(G) = \{s(Tr) \mid Tr \in T(G)\}$ the language generated by G, through the set of DR-trees.

Let $Tr_1 = (Nod_1, Ed_1, Rt_1, Ann_1)$ and $Tr_2 = (Nod_2, Ed_2, Rt_2, Ann_2)$ be two DR- trees created by the grammar $G = (N, T, S, P)$. We say that Tr_1 and Tr_2 are **DR-equivalent** iff there is an isomorphism $f : Nod_1 \to Nod_2$ between Tr_1 and Tr_2 as V-annotated trees and:

1. $f(i, j) = (s, j)$, for any node $(i, j) \in Nod_1$.
2. If $(i, j) \in Nod_1$, with $j \neq 1$ and $f(i, j) = (s, j)$, then:
   a. if there is exactly one node $n \in Nod_1$ such that $Ed(n) = (i, j)$ then $f(n) = (s, j - 1)$.
   b. if there are exactly two nodes $n_1, n_2 \in Nod$ such that $Ed(n_1) = Ed(n_2) = (i, j)$ then either:
      b1. if $n_1 = (i, k)$ and $n_2 = (l, m)$ with $l > i$, then $f(n_1) = (s, k)$ and $f(n_2) = (t, m)$ with $t > s$ or
      b2. if $n_1 = (l, k)$ and $n_2 = (i, m)$ with $l < i$, then $f(n_1) = (t, k)$ and $f(n_2) = (s, m)$ with $t < s$.

We say that f is a **DR-isomorphism** between Tr_1 and Tr_2.

Let T be an alphabet and *Tr = (Nod, Ed, Rt, Ann)* be a T-annotated tree such that $Nod \subseteq Nat$. We call Tr a **D-tree over** T (see an example of D-tree in Figure 1).

We say that Tr is **complete** iff for any leaf $i \in Nod$, if $i > 1$ then also $i - 1 \in Nod$.

For any complete D-tree *dTr = (dNod, dEd, dRt, dAnn)* over T, we define the **sentence** associated with dTr by $s(dTr) = a_1 a_2 \ldots a_n$, where $n = \max\{i \mid i \in dNod\}$ and $dAnn(i) = a_i$, for any $i \in [n]$. Obviously, $s(Tr) \subseteq T^+$.

Let $Tr_1 = (Nod_1, Ed_1, Rt_1, Ann_1)$ and $Tr_2 = (Nod_2, Ed_2, Rt_2, Ann_2)$ be two D-trees over an alphabet T. We say that Tr_1 and Tr_2 are **D-equivalent** iff there is an isomorphism $f : Nod_1 \to Nod_2$ between Tr_1 and Tr_2 as T-annotated trees such that if $i, j \in Nod_1$ are two nodes with $Ed_1(j) = i$ then $(i - j)(f(i) - f(j)) > 0$. We say that f is a **D-isomorphism** between Tr_1 and Tr_2.

Let *Tr = (Nod, Ed, Rt, Ann)* be a DR-tree. We may transform Tr in a D-tree *dTr = (dNod, dEd, dRt, dAnn)* in the following way:

1. $dNod = \{i \mid (i,1) \in Nod\}$.
2. $dRt = i$ iff $Rt = (i,j)$, for some $j \in Nat$.
3. Let $i \in dNod$ be a node in dTr and $(i,1) \in Nod$ the corresponding leaf in Tr. We consider the path $p = (n_1, n_2, \ldots, n_k)$ in Tr from $n_1 = (i,1)$ to $n_k = Rt$ (cf. Proposition 2.1). We also consider the natural number $r = \max\{l \mid n_l = (i,j), j \in Nat\}$. Then one of the following cases necessarily occurs:
   a. If $r = k$ then $dRt = i$.
   b. If $r < k$ and $n_{r+1} = (s,t)$ then $dEd(i) = s$.
4. $dAnn(i) = Ann(i,1)$, for any $i \in dNod$.

We say that dTr is the D-tree **corresponding** to Tr. The D-tree dTr represented in Figure 1 is corresponding to the DR-tree Tr from the same figure.

Remark 2.3 *If Tr is a complete DR-tree, then the corresponding D-tree dTr is also complete.*

Let $G = (N, T, S, P)$ be a dependency grammar. We denote by:

- *dT(G)* the set of D-trees corresponding to $T(G)$;
- *D-L(G)* $= \{s(dTr) \mid dTr \in dT(G)\}$ the language generated by G, through the set of D-trees.

If dTr is a D-tree corresponding to a DR-tree created by a dependency grammar G, we say that dTr is created by G as well. Obviously, *DR-L(G)=D-L(G)*, for any dependency grammar G.

Proposition 2.4 *Let Tr_1 and Tr_2 be two DR-equivalent DR-trees. Let dTr_1 and respectively dTr_2 the two corresponding D-trees. Then dTr_1 and dTr_2 are D-equivalent.*

Proof We consider $f : Nod_1 \to Nod_2$ a DR-isomorphism between $Tr_1 = (Nod_1, Ed_1, Rt_1, Ann_1)$ and $Tr_2 = (Nod_2, Ed_2, Rt_2, Ann_2)$. We take the two D-trees corresponding to Tr_1 and Tr_2, $dTr_1 = (dNod_1, dEd_1, dRt_1, dAnn_1)$, respectively $dTr_2 = (dNod_2, dEd_2, dRt_2, dAnn_2)$.

We define $f' : dNod_1 \to dNod_2$ by

$$f'(i) = i' \iff f(i,1) = (i',1),$$

for any $i \in dNod_1$. From the construction of the D-trees dTr_1 and dTr_2 and from Proposition 2.3, it follows that the definition of f' is correct and also that f' is a bijection from $dNod_1$ to $dNod_2$. We will check the conditions in which f' would establish an equivalence between dTr_1 and dTr_2.

1. Let $i \in dNod_1$ be a node. If $i \neq Rt$ there is a node $s \in dNod_1$ such that $dEd_1(i) = s$. From the construction of dTr_1 it results that there are two nodes $(i,j),(s,t) \in Nod_1$ such that $Ed_1(i,j) = (s,t)$. We have that

   $$f'(s) = s' \iff f(s,1) = (s',1)$$

   and

   $$f'(i) = i' \iff f(i,1) = (i',1).$$

   Since f is a bijection it follows that $i' \neq s'$. It results that $dEd_2(i') = s'$. We have that

   $$(s',t) = f(s,t) = f(Ed_1(i,j)) = Ed_2(f(i,j)) = Ed_2(i',j)$$

   hence

   $$f'(dEd_1(i)) = f'(s) = s' = dEd_2(i') = dEd_2(f'(i)).$$

2. We have $dRt_1 = i$ iff there exists $j \in Nat$ such that $Rt_1 = (i,j)$. We also have $f(Rt_1) = Rt_2$. Then

   $$f'(dRt_1) = i' \iff f(i,1) = (i',1) \iff f(i,j) = (i',j) \iff$$

   $$\iff Rt_2 = (i',j) \iff dRt_2 = i'.$$

   It results that $f'(dRt_1) = dRt_2$.

□

Let $Tr = (Nod, Ed, Rt, Ann)$ be an annotated tree, $n \in Nod$ be a node. We define the **covering subtree** of n in Tr by the following annotated tree, $Tr_n = (Nod_n, Ed_n, Rt_n, Ann_n)$ such that

- $Nod_n = \{n' \mid \text{there is a path from } n' \text{ to } n \text{ in } Tr\}$.

- $Ed_n(n') = Ed(n')$, for any $n' \in Nod_n \setminus \{n\}$.
- $Rt_n = n$;
- $Ann_n(n') = Ann(n')$, for any $n' \in Nod_n$.

Let $Tr = (Nod, Ed, Rt, Ann)$ be a complete DR-tree, $n \in Nod$ be a node and $Tr_n = (Nod_n, Ed_n, Rt_n, Ann_n)$ be the covering subtree of n in Tr. We define the **coverage** of n in Tr by the set:

$$Cov(n, Tr) = \{i \in Nat \mid \text{ there is a node } (i, 1) \in Nod_n\}.$$

Let $n \in Nod$ be a node in Tr such that $Cov(n, Tr) = \{i_1, i_2, \ldots, i_m\}$, with $i_1 < i_2 < \ldots < i_m$ and $i_{j+1} - i_j > 1$ for some $j \in Nat$, $j < m$. We say that the pair (i_j, i_{j+1}) is a **gap** in Tr at the node n.

Let $Tr = (Nod, Ed, Rt, Ann)$ be a complete DR-tree, $n \in Nod$ be a node and $Cov(n, Tr)$ be its coverage. The symbol *DR-Ng(n,Tr)* represents the number of gaps in Tr at the node n. The symbol *DR-Ng(Tr)* is the maximum number of gaps in Tr at any node $n \in Nod$:

$$\textit{DR-Ng(Tr)} = \max\{\textit{DR-Ng(n,Tr)} \mid n \in Nod\}.$$

We say that *DR-Ng(Tr)* is the **DR-node-gaps complexity**) of Tr.

We say that Tr is **projectively parsed** (or simply **projective**) iff *DR-Ng(Tr)=0*.

Let $G = (N, T, S, P)$ be a dependency grammar. We denote by:

- $T(G, i) \subseteq T(G)$ the set of complete and rooted by S DR-trees Tr created by G with at most i gaps, *DR-Ng(Tr)* $\leq i$;
- *DR-L(G,i)* $= \{s(Tr) \mid Tr \in T(G, i)\}$ the language generated by G, through DR-trees with at most i gaps.

We mention the following obvious claim.

Claim 1 *Let G be a dependency grammar. Then the following inclusions hold.*

i) $T(G, i) \subseteq T(G, i + 1) \subseteq T(G)$, *for any* $i \in Nat_0$.

ii) *DR-L(G,i)* $\subseteq$ *DR-L(G,i+1)* $\subseteq$ *DR-L(G)*, *for any* $i \in Nat_0$.

Proposition 2.5 *For any complete DR-tree Tr_1 created by a dependency grammar G there exists a DR-equivalent projective complete DR-tree Tr_2 created by the same grammar G.*

Proof Let $Tr_1 = (Nod_1, Ed_1, Rt_1, Ann_1)$ be a DR-tree and let us suppose that $DR\text{-}Ng(Tr_1) > 0$. Let $n_1 = (i, j)$ be a node in Nod_1, such that $DR\text{-}Ng(n_1, Tr_1) > 0$ and for all the nodes $n' = (i', j') \in Nod_1$ such that $i' \in Cov(n_1, Tr_1)$, we have $DR\text{-}Ng(n', Tr_1) = 0$.

It results that there are two nodes $n_2 = (k, l)$ and $n_3 = (i, m)$ such that $Ed_1(n_2) = Ed_1(n_3) = n_1$. Let us suppose that $k < i$ (the case $i < k$ can be solved analogously). Let us denote

$$Cov(n_p, Tr_1) \;=\; \{s_1^p, s_2^p, \ldots, s_{t_p}^p\},$$

for $p = \{1, 2, 3\}$. We have that $t_1 = t_2 + t_3$ and since $DR\text{-}Ng(n_2, Tr_1) = DR\text{-}Ng(n_3, Tr_1) = 0$ we have

$$s_t^1 \;=\; \begin{cases} s_t^2, & \text{if } t \leq t_2 \\ s_{t-t_2}^3, & \text{if } t > t_2 \end{cases}$$

It follows that Tr_1 has only one gap at the node n_1, which is between $s_{t_2}^1$ and $s_{t_2+1}^1$, i.e. $DR\text{-}Ng(n_1, Tr_1) = 1$.

Denote by $s = s_1^3 - s_{t_2}^2 - 1$ the number of nodes, which are between the coverage of n_2 and the coverage of n_3 and are not in the coverage of n_1.

We define an application $f : Nod_1 \to Nat \times Nat$ by:

$$f(p, r) = \begin{cases} (p, r) & \text{if } p < s_1^1 \text{ or } p \geq s_1^3, \\ (p + s, r) & \text{if } s_1^2 \leq p \leq s_{t_2}^2, \\ (p - t_2, r) & \text{if } s_{t_2}^2 < p < s_1^3, \end{cases}$$

We observe that the co-restriction of f to $f(Nod_1)$ is a bijection between Nod_1 and $f(Nod_1)$. We define a DR-tree $Tr_2 = (Nod_2, Ed_2, Rt_2, Ann_2)$ such that

- $Nod_2 = f(Nod_1)$.
- $Ed_2(f(n)) = f(Ed(n))$, for any $n \in Nod_1$.
- $Rt_2 = f(Rt_1)$;
- $Ann_2(f(n)) = Ann(n)$, for any $n \in Nod_1$.

It is easy to see that:

- the definition of Tr_2 is correct,
- Tr_2 is also a complete DR-tree created by the grammar G and
- f is a DR-isomorphism between Tr_1 and Tr_2.

Moreover, we observe that $DR\text{-}Ng(f(n_1), Tr_2) = 0$, while the node-gaps-complexity of Tr_1 was preserved under the isomorphism in all other nodes, excepting n_1.

It results that, under this isomorphism, the number of nodes with gaps decreased at least by one. By repeating the transformation for a finite number of times, we obtain a projective complete DR-tree created by G and DR-equivalent with the initial DR-tree.

□

Now, let us consider the same node-gaps-complexity measure for D-trees.

Let $dTr = (dNod, dEd, dRt, dAnn)$ be a complete D-tree, $n \in dNod$ be a node and $dTr_n = (dNod_n, dEd_n, dRt_n, dAnn_n)$ be the covering subtree of n in dTr. We define the **coverage** of n in dTr by the set:

$$Cov(n, dTr) = dNod_n.$$

Let $n \in dNod$ be a node in dTr such that $Cov(n, dTr) = \{i_1, i_2, \ldots, i_m\}$, with $i_1 < i_2 < \ldots < i_m$ and $i_{j+1} - i_j > 1$ for some $j \in Nat$, $j < m$. We say that the pair (i_j, i_{j+1}) is a **gap** in dTr at the node n.

Let $dTr = (dNod, dEd, dRt, dAnn)$ be a complete D-tree, $n \in dNod$ be a node and denote by $Cov(n, dTr)$ its coverage. The symbol $D\text{-}Ng(n, dTr)$ represents the number of gaps in dTr at the node n. The symbol $D\text{-}Ng(dTr)$ is the maximum number of gaps in dTr at any node $n \in dNod$:

$$D\text{-}Ng(dTr) = \max\{D\text{-}Ng(n, dTr) \mid n \in dNod\}.$$

We say that $D\text{-}Ng(dTr)$ is the **D-node-gaps complexity** of dTr.

We say that dTr is **projective** iff $D\text{-}Ng(dTr) = 0$.

Let $G = (N, T, S, P)$ be a dependency grammar. We denote by:

- $dT(G, i)x = \{dTr \mid Tr \in T(G), D\text{-}Ng(dTr) \leq i\}$ the set of D-trees dTr created by G with at most i gaps;
- $D\text{-}L(G, i) = \{s(dTr) \mid dTr \in dT(G, i)\}$ the language generated by G, through D-trees with at most i gaps.

We can establish the following result between the node-gaps-complexity of a DR- tree and the same measure of the corresponding D-tree.

Lemma 2.6 *Let Tr be a complete DR-tree and dTr be the corresponding D-tree. If n is a node in dTr, then there exists n' a node in Tr such that $Cov(n, dTr) = Cov(n', Tr)$.*

Proof Let $Tr = (Nod, Ed, Rt, Ann)$ be a complete DR-tree and $dTr = (dNod, dEd, dRt, dAnn)$ be the corresponding D-tree. Let $n = i \in Nat$ be a node in dTr. From the construction of dTr from Tr it follows that there is at least one node (i, j) in Tr, with $j \in Nat$. Let us consider $n' = (i, j')$, where

$$j' = \max\{j \mid (i, j) \in Nod\}.$$

We will prove that $Cov(n, dTr) = Cov(n', Tr)$.

It is easy to prove, following the construction of dTR from Tr that if there is a path between two nodes (p, r) and (s, t) in Tr then there is also a path between the nodes p and s in dTr. Conversely, if there is a path between two nodes p and s in dTr then there exists a node (s, t) in Tr (which can be taken as (s, t') with $t' = \max\{t \mid (s, t) \in Nod\}$) such that for any node (p, r) in Tr, there is a path from (p, r) to (s, t).

It follows that for any $k \in dNod$ (equivalent with: for any $(k, 1) \in Nod$) there is a path from k to i in dTr if and only if there is a path from $(k, 1)$ to (i, j') in Tr, which completes the proof.

□

Proposition 2.7 *If Tr is a complete DR-tree with DR-Ng$(Tr) = k$, then dTr, the corresponding D-tree has D-Ng$(dTr) \leq k$.*

Proof It follows from Lemma 2.6. If dTr has a node n with j gaps, then Tr has also a node n' with j gaps. This implies that

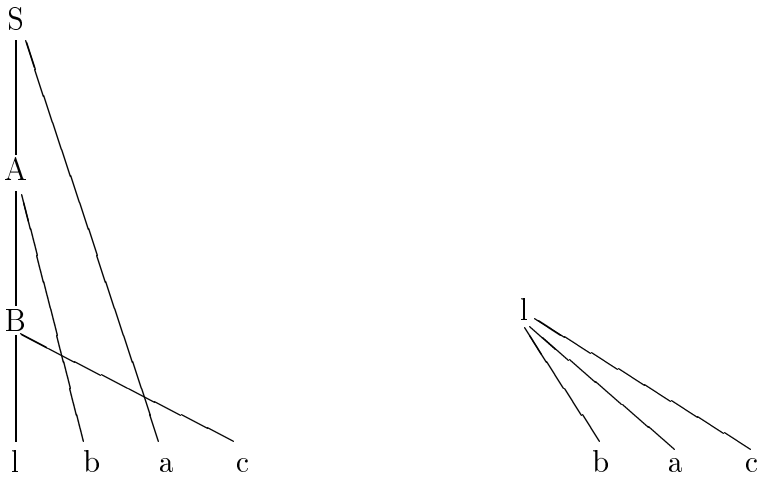
$$\textit{D-Ng(dTr)} \leq \textit{DR-Ng}(Tr).$$

□

Corollary 2.8 *If Tr is a projective complete DR-tree, then dTr, the corresponding D-tree is also projective.*

Proof It follows immediately from Proposition 2.7.

□



a. DR-tree Tr b. D-tree dTr

Figure 2: Projective D-tree corresponding to a non-projective DR-tree created by the dependency grammar in Example 2.1

Remark 2.4 *The reverse statement is not true as we may see in the below example.*

Example 2.1 *Let us consider* $G = (N, T, S, P)$ *a dependency grammar, with* $N = \{A, B, C, S\}$, $T = \{a, b, c, l\}$, $P = \{S \to Aa, S \to l, A \to_L Bb, B \to_L Cc, B \to_L lc, C \to_L Aa\}$. *We can easily define a non-projectively parsed DR-tree for which the corresponding D-tree is projectively parsed (see Figure 2).*

Moreover, we can establish the following results between the languages generated by a dependency grammar through DR-trees and D-trees with the same maximal number of gaps.

Corollary 2.9 *Let* G *be a dependency grammar and* $i \in Nat_0$ *be a natural number. Then DR-L*$(G, i) \subseteq$ *D-L*(G, i).

Proof It results from the definitions of *DR-L(G,i)*, *D-L*(G,i) and from Proposition 2.7.

□

3 D-trivial left dependency grammars

In this section, we will define and study several particular forms of DR-trees and dependency grammars.

Definition 1 Let G be a dependency grammar. We say that G is a **D-trivial** dependency grammar if G creates only projectively-parsed D-trees, i.e. *D-Ng*$(dTr) = 0$, for any D-tree dTr created by G.

Definition 2 Let $G = (N, T, S, P)$ be a dependency grammar and let us denote $V = N \cup T$. We say that G is a **left** dependency grammar if G does not contain productions of the form $A \rightarrow_R BC$, i.e. $P \subseteq (N \times (V \setminus \{S\})(V \setminus \{S\}) \times \{L\}) \cup (\{S\} \times T)$.

Remark 3.1 *Considering the notations from the above definition, let us remark that productions of the form* $S \rightarrow a$, *with* $a \in T$, *may appear only in DR-trees created by* G *that have the form* $Tr = (Nod, Ed, Rt, Ann)$ *with:*

- $Nod = \{(1,1),(1,2)\}$.
- $Ed(1,1) = (1,2)$.
- $Rt = (1,2)$;
- $Ann(1,2) = S$ *and* $Ann(1,1) = a$.

Let us also observe that the dependency grammar defined in Example 2.1 is a D-trivial left dependency grammar.

The following result characterizes the form of DR-trees created by a D-trivial left dependency grammar.

Lemma 3.1 *Let* $Tr = (Nod, Ed, Rt, Ann)$ *be a complete DR-tree created by a D-trivial left dependency grammar* $G = (N, T, S, P)$. *The following properties hold:*

a) Let $n_1 = (i,j) \in Nod$ *be a node in* Tr. *Let* Tr_{n_1} *be the covering subtree of* n_1 *in* Tr. *Then* $(k,l) \in Tr_{n_1}$ *implies* $i \leq k$.

b) Denote $k = \max\{i \mid \exists j > 1, (i,j) \in Nod\}$. *If* $(k,l) \in Nod$ *then* $(k,j) \in Nod$, *for any* $1 \leq j \leq l$.

c) Denote $l = \max\{j \mid (k,j) \in Nod\}$. *Then for any node* $(i,j) \in Nod$, *with* $i < k$, $j > 1$ *implies* $j = k - i + l$.

d) $Rt = (1, k + l - 1)$.

e) Denote $m = \max\{i \mid \exists j \in Nat, (i,j) \in Nod\}$. *Then* $m = k + l - 1$.

Proof

a) Suppose to a contradiction that there exists $n_2 = (k,l) \in Tr_{n_1}$ such that $i > k$. Let us consider the path $Path(n_2, n_1)$ which exists because of the definition of a covering subtree. It is easy to see that this path should contain at least two consecutive nodes $n_3 = (p,r)$ and $n_4 = (s,t)$ such that $p < r$. We also have that $Ed(n_3) = n_4$, hence it should exists a production $A \rightarrow_R BC \in P$ such that $Ann(n_4) = A$ and $Ann(n_3) = B$. Contradiction, since G does not contain such productions.

b) For $l \leq 2$ the property is D-trivial. Let $n_1 = (k,l) \in Nod$, with $l > 2$ be a node. From the definition of a DR-tree, it follows that there exists at least a node $n_2 = (m, l-1) \in Nod$ such that $Ed(n_2) = n_1$. From the property a) of this lemma, it follows that $m \geq k$. Since $(i,j) \in Nod$, $i > k$ implies $j = 1$ and since $m \geq k$ and $l - 1 > 1$, we obtain $m = k$. We proved that $(k,l) \in Nod$ implies $(k, l-1) \in Nod$, hence also the property b) is true.

c) Let us consider a node $n_1 = (k-1, j) \in Nod$, with $j > 1$. Suppose to a contradiction that $j \leq l$. From the definition of a left dependency grammar, it results that this node should have two daughters, $n_2 = (k-1, r)$ and $n_3 = (s,t)$. From property a), it follows that $s > k - 1$. But $s \neq k$, because the node (k,l) cannot be a daughter of the node n_1 (since $j \leq l$). It results that $s > k$, hence $t = 1$. The node (k,l) also should have a daughter $n_4 = (u, 1)$, with $u > k$. Let us suppose that $s < u$ (if this is not true, we can permute nodes n_3 and n_4, keeping the DR-isomorphism through out the transformation and we obtain still a complete DR-tree created by G). Transforming the DR-tree in a D-tree $dTr = (dNod, dEd, dRt, dAnn)$ we observe

that $k < s < u$, $k, u \in Cov(k, dTR)$, while $s \notin Cov(k, dTR)$, thus $D\text{-}Ng(k, dTr) \geq 1$, hence $D\text{-}Ng(dTr) \geq 1$, which is a contradiction with the initial assumption that G creates only projective D-trees.

It results that $j > l$. Denote $v = \min\{j \mid (k-1, j) \in Nod, j > l\}$ and let $(k-1, v)$ be a node. Again it results that n_1 should have two daughters, $(k-1, r)$ and (s, t), with $s > k-1$. It results that $r = 1$. From the fact that $l = \max\{j \mid \exists i \geq k, (i, j) \in Nod\}$, we obtain that $t \leq l$. Than, from the definition of a DR-tree, it results that $t = l$, $v = l + 1$ and $Ed(k, l) = (k-1, l+1)$.

Now, suppose to contradiction that there exists another node $(k-1, j) \in Nod$, with $j > l + 1$. We take $y = \min\{j \mid (k-1, j) \in Nod, j > l+1\}$ and it follows as above that $y = l + 2$. Following a similar reasoning as for the case $j \leq l$ (now the node (k, l) cannot be the daughter of the node $(k-1, l+2)$, because it already is the daughter of the node $(k-1, l+1)$) we obtain a contradiction with the assumption that G creates only projective D-trees. It results that the only possible node $(k-1, j) \in Nod$ with $j > 1$ is $(k-1, l+1)$. But for this node the condition $l + 1 = k - (k-1) + l$ is satisfied.

We can apply the same reasoning decreasing by step 1 from $k-1$ to 1, for any i between 1 and $k-1$ and prove in this way the property c).

d) It follows from properties a) and c).

e) From the above properties, it results that for any node $(i, 1)$ with $i > k$ there is exactly one node (k, j) with $1 < j \leq l$ such that $Ed(i, 1) = (k, j)$. As there are $l - 1$ nodes (k, j) with $j > 1$, it results that there are also exactly $l - 1$ nodes $(i, 1)$ with $i > k$, thus $m = k + l - 1$.

□

We say that a DR-tree created by a D-trivial left dependency grammar is a **D-trivial left DR-tree**. Using notations from Lemma 3.1, we say that a D-trivial left DR-tree Tr is a **DR left bush** if $k = 1$ and a **DR left path** if $l = 2$. Let us note that these notations are derived from the shape of corresponding D-trees (see Figure 3).

Lemma 3.2 *Let Tr be a complete D-trivial left DR-tree with m leaves. Then the longest path in Tr has exactly m nodes.*

a. Left path b. Left bush

Figure 3: D-trees created by a DR left path, respectively a DR left bush

Proof Using notations from Lemma 3.1, it is easy to see that the longest path in Tr starts from the node (k, l) and ends with the root Rt of Tr. This path includes $l + k - 1$ nodes. But $m = l + k - 1$ (property e) from the same lemma.

□

The following result characterizes the form of D-trivial left dependency grammars.

Proposition 3.3 *Let $G = (N, T, S, P)$ be a D-trivial left dependency grammar. Then there exists a partition of the set of nonterminals $N = N_1 \cup N_2 \cup N_3$ such that $N_i \cap N_j = \emptyset$, for any $1 \leq i < j \leq 3$ and the set of productions:*

$$\begin{aligned} P \subseteq\ & ((N_1 \cup N_2) \times T(N \setminus \{S\}) \times \{L\}) \cup \\ & ((N_2 \cup N_3) \times (N_3 \setminus \{S\})T \times \{L\}) \cup \\ & (N \times TT \times \{L\}) \cup \\ & (\{S\} \times T). \end{aligned}$$

Proof We define the decomposition of N in the following way:

$$\begin{aligned} N_1 &= \{A \mid \exists(A, aB, L) \in P, \nexists(A, Ba, L) \in P, A, B \in N, a \in T\} \\ N_2 &= \{A \mid \exists(A, aB, L) \in P, \exists(A, Ba, L) \in P, A, B \in N, a \in T\} \\ N_3 &= \{A \mid \nexists(A, aB, L) \in P, A, B \in N, a \in T\} \end{aligned}$$

Obviously, N_1, N_2, N_3 form a partition of N (they are disjoint and their union is exactly N). From the above definitions, we observe that G cannot have productions of the form:

- $A \rightarrow_L aB$, with $A \in N_3$, $B \in N$, $a \in T$,
- $A \rightarrow_L Ba$, with $A \in N_1$, $B \in N$, $a \in T$.

From Lemma 3.1, we observe that G cannot have productions of the form:

- $A \rightarrow_L BC$, with $A, B, C \in N$,
- $A \rightarrow_L Ba$, with $A \in N_2 \cup N_3$, $B \in N_1 \cup N_2$, $a \in T$,

because such types of productions would create non-D-trivial left DR-trees.

These observations complete the proof.

□

4 An infinite sequence of classes of languages

In the beginning of this section, we introduce the main classes of languages which we address in our paper. We denote by:

- *DR-*$\mathcal{L}$ the class of languages generated by all dependency grammars through the set of DR-trees;
- *DR-*$\mathcal{L}(i)$ the class of languages generated by all dependency grammars through DR-trees with at most i gaps;
- *tlDR-*$\mathcal{L}$ the class of languages generated by all D-trivial left dependency grammars through the set of DR-trees;

- *tlDR-$\mathcal{L}(i)$* the class of languages generated by all D-trivial left dependency grammars through DR-trees with at most i gaps.

We also introduce two types of languages of a particular kind. Let $n \in Nat$ be a natural number, $V = \{b_1, \ldots, b_n\}$ be an alphabet and $l \notin V$ be a distinct symbol. Denote $L_{lb_1 \ldots b_n}$ and $L^{total}_{lb_1 \ldots b_n}$ two languages over $V \cup \{l\}$ by:

$$\begin{aligned} L_{lb_1 \ldots b_n} &= \{l(b_1)^m \ldots (b_n)^m \mid m \in Nat_0\}, \\ L^{total}_{lb_1 \ldots b_n} &= \{lw \in V^+ \mid |w|_{b_1} = \ldots = |w|_{b_n}\}, \end{aligned}$$

where $|w|_b$ denotes the number of occurrences of the symbol b in the string w.

Let us observe that the grammar G in Example 2.1 generates the language L^{total}_{labc}.

Proposition 4.1 *Let $i \in Nat_0$ be a natural number, $V = \{b_1, \ldots, b_{2i+1}\}$ be an alphabet and $l \notin V$ be a distinct symbol. Then there exists a language L over $V \cup \{l\}$ such that $L_{lb_1 \ldots b_{2i+1}} \subseteq L \subseteq L^{total}_{lb_1 \ldots b_{2i+1}}$ and $L \in tlDR - \mathcal{L}(i)$.*

Proof Consider $G = (N, V \cup \{l\}, B_{2i+1}, P)$ a D-trivial left dependency grammar such that $N = \{B_j \mid j \in [2i+1]\}$, $P = \{B_{j+1} \to B_j b_j \mid j \in [2i]\} \cup \{B_1 \to B_{2i+1} b_{2i+1}, B_{2i+1} \to l\}$. We take L = DR-L$(G, i)$, hence $L \in tlDR - \mathcal{L}(i)$.

It is easy to observe that $L \subseteq L^{total}_{lb_1 \ldots b_{2i+1}}$.

Let $m \in Nat_0$ be a natural number and $Tr_m = (Nod_m, Ed_m, Rt_m, Ann_m)$ be a D-trivial left Dr-tree such that

- $Nod_m = \{(1, t+1) \mid t \in [(2i+1)m+1]\} \cup \{(s, 1) \mid s \in [(2i+1)m+1]\}$.
- $$Ed_m(s,t) = \begin{cases} (1, t+1), & \text{if } s = 1; \\ (x + (2i+1)(y-1) + 2, 1) & \text{if } s = (x-1)m + y + 1, \\ & t = 1, x = 2(z-1)+1, \\ & z \in [i+1], y \in [m]; \\ (x + (2i+1)(m-y) + 2, 1) & \text{if } s = (x-1)m + y + 1, \\ & t = 1, x = 2z, \\ & z \in [i+1], y \in [m], \end{cases}$$
  for any $(s, t) \in Nod_m$.
- $Rt_m = (1, (2i+1)m + 2)$;

- $$Ann_m(s,t) = \begin{cases} l, & \text{if } s=1, t=1; \\ B_{2i+1} & \text{if } s=1, t=2; \\ B_x & \text{if } s=1, t=x+(2i+1)(y-1)+2, \\ & x \in [2i+1], y \in [m]; \\ b_x & \text{if } s=(x-1)m+y+1, t=1, \\ & x \in [2i+1], y \in [m], \end{cases}$$

  for any $(s,t) \in Nod_m$.

We can count the number of gaps of any node (s,t) in Tr_m in the following way:

$$DR\text{-}Ng((s,t), Tr_m) = \begin{cases} z, & \text{if } s=1, t=2z+2 \text{ or } t=2z+3, z \in [i]; \\ i, & \text{if } s=1, 2i+4 \leq t \leq (2i+1)(m-1)+3; \\ i-z, & \text{if} s=1, t=(2i+1)(m-1)+2z+2 \text{ or} \\ & t=(2i+1)(m-1)+2z+3, z \in [i-1]; \\ 0 & \text{otherwise.} \end{cases}$$

We have that $DR\text{-}Ng(Tr_m) = \max\{DR\text{-}Ng(no, Tr_m) \mid no \in Nod_m\} = i$, which proves that Tr_m is D-trivial left DR-tree with at most i gaps, created by G. Thus $l(b_1)^m \dots (b_n)^m = s(Tr_m) \in L$, for any $m \in Nat \cup \{0\}$, hence $L_{lb_1 \dots b_{2i+1}} \subseteq L$.

□

Lemma 4.2 (pumping lemma) *Let $L \in tlDR\text{-}\mathcal{L}(i)$ be a language. Then, there exist two natural constants $p, r \in Nat$ such that for any sentence $w \in L$ with $|w| > p$, there exists a decomposition of w in $w = \alpha_1 a_1 \alpha_2 \dots \alpha_r a_r \alpha_{r+1}$ such that the following conditions hold:*

i) $|a_h| > 0$, *for any* $h \in [r]$;

ii) $|a_1 \dots a_r| < p$;

iii) $\alpha_1 (a_1)^j \alpha_2 \dots \alpha_r (a_r)^j \alpha_{r+1} \in L$, *for any* $j \in Nat_0$.

iv) If $r > i+1$ *there are at most* i *distinct indices* h *such that* $1 < h < r+1$ *and* $|\alpha_h| > p$.

Proof Let $G = (N, T, S, P)$ be a D-trivial left dependency grammar such that $L = DR\text{-}L(G,i)$. Let n be the number of nonterminals in G. Denote $p = n + 1$. Let us consider a sentence $w \in L$ with $|w| > p$ and a left D-trivial DR-tree Tr created by G such that $s(Tr) = w$. Since Tr has exactly $|w|$ leaves, from Lemma 3.2, we have that there exists at least a path $(n_1, n_2, \ldots, n_{|w|})$ in Tr (the longest path in Tr). Since $|w| > n + 1$, nodes on this path are annotated by more than n nonterminals, thus in the sequence $Ann(n_2), \ldots, Ann(n_{|w|})$ there exist at least two equal nonterminals. Let us consider the first two equal nonterminals in this sequence, i.e. let us consider two indices $1 < s < t \leq |w|$ such that $Ann(n_s) = Ann(n_t)$ and all nonterminals in the sequence $Ann(n_2), \ldots, Ann(n_{t-1})$ are distinct.

In the following, we will use for Tr notations from Lemma 3.1. There are three possible cases:

1. $1 < s < t \leq l$. We consider the sequence of nodes $n_t = (k, t), n_{t-1} = (k, t-1), \ldots, n_{s+1} = (k, s+1)$. Each of these nodes has a left dominated daughter, which is a leaf. Consider this sequence of leaves, $(no_1, 1), \ldots, (no_{t-s}, 1)$ such that $Ed(no_i, 1) = n_{s+i}$, for any $1 \leq i \leq t-s$. We denote by a_h, with h from 1 to a certain r, a sequence of terminals $Ann(no_x, 1)Ann(no_{x+1}, 1) \ldots Ann(no_{x+y}, 1)$, where $no_x, no_{x+1}, \ldots, no_y$ is a maximal sequence of consecutive natural numbers. Consider the decomposition $w = \alpha_1 a_1 \alpha_2 \ldots \alpha_r a_r \alpha_{r+1}$. We have $|a_1 \ldots a_r| = r$. Since the path from n_{s+1} to n_t has at least one node and at most n nodes (as all nonterminals on this path are distinct), we obtain $r < p$, which, together with $|a_h| > 0$, for any $h \in [r]$ fulfills conditions i) and ii) of the pumping lemma. We may replace the covering subtree Tr_{n_t} with the covering subtree Tr_{n_s} (preserving the completeness under the transformation) and the resulted DR-tree Tr_0 is still a complete DR-tree created by G. Obviously the transformation will not increase the number of gaps. It results $s(Tr_0) = \alpha_1 \alpha_2 \ldots \alpha_{r+1} \in L$, which proves condition iii) of the lemma for $j = 0$. For $j = 1$, $\alpha_1 a_1 \alpha_2 \ldots \alpha_r a_r \alpha_{r+1} = w$, which obviously belongs to L. Moreover, we may replace the covering subtree Tr_{n_s} with the covering subtree Tr_{n_t} in such a way that new introduced leaves corresponding to a string a_h, $h \in [r]$ will stick immediately after the leaves from the initial covering subtree Tr_{n_s} corresponding to the same sequence a_h. The resulted DR-tree Tr_2 is still a complete DR-tree created by G. Again the transformation will not increase the number of gaps. We can repeat this transformation for an unlimited number

of times, obtaining in this way an infinite sequence Tr_j, with $j \geq 2$, of complete DR-trees created by G with at most i gaps. It results that $s(Tr_j) = \alpha_1(a_1)^j\alpha_2 \ldots \alpha_r(a_r)^j\alpha_{r+1} \in L$, for any $j \geq 2$ which completes condition iii) of the lemma.

Finally, if $r > i+1$, suppose towards a contradiction that there exist $i+1$ distinct indices h_j such that $1 < h_j < r+1$ and $|\alpha_{h_j}| > p$, for all $j \in [i+1]$. We observe that none of the gaps induced by α_{h_j}, with $j \in [i+1]$ in the coverage of n_t in Tr can be fulfilled by the coverage of n_s in Tr, since the latest one has at most p elements. It follows that $DR\text{-}Ng(n_t, Tr) \geq i+1$ which is a contradiction with the assumption that $Tr \in T(G, i)$. This means that also the condition iv) of the lemma is true.

2. $1 < s < l < t$. If we replace the covering subtree Tr_{n_s} with the covering subtree Tr_{n_t} in a similar way as above, we obtain a DR-tree created by G, but which is not a D-trivial left DR-tree. This contradicts the fact that G is a D-trivial left dependency grammar, which creates only D-trivial left DR-trees. Thus, this case is not possible under the pumping lemma's assumptions.

3. $l \leq s < t \leq |w|$. We proceed in a similar way as for the first case, by taking $r = 1$. Condition iv) of the lemma does not apply in this case.

□

Proposition 4.3 *Let $i, k \in Nat_0$ be two natural numbers such that $i < k$, $V = \{b_1, \ldots, b_{2k+1}\}$ be an alphabet, $l \notin V$ be a distinct symbol and L be a language over $V \cup \{l\}$ such that $L_{lb_1 \ldots b_{2k+1}} \subseteq L \subseteq L^{total}_{lb_1 \ldots b_{2k+1}}$. Then $L \notin tlDR\text{-}\mathcal{L}(i)$.*

Proof Suppose towards a contradiction that $L \in tlDR{-}\mathcal{L}(i)$. It follows that the pumping lemma holds for L and. Consider p and r the two constants from the pumping lemma, a natural number n such that $n > 2p$ and the sentence $w = l(b_1)^n \ldots (b_{2k+1})^n$. We have $|w| = (2k+1)n + 1 > p$, hence, from the pumping lemma, it should exists a decomposition of w in $w = \alpha_1 a_1 \alpha_2 \ldots \alpha_r a_r \alpha_{r+1}$ such that conditions i)-iv) of the lemma hold. Take $j = 0$ in condition iii). It results that $w_0 = \alpha_1 a_1 \alpha_2 \ldots \alpha_r a_r \alpha_{r+1} \in L$. Since $L \subseteq L^{total}_{lb_1 \ldots b_{2k+1}}$, it follows that $|w_0|_{b_1} = \ldots = |w_0|_{b_{2k+1}}$ and further that also $|a_1 \ldots a_r|_{b_1} = \ldots = |a_1 \ldots a_r|_{b_{2k+1}}$. Since $|a_1 \ldots a_r| < p$ and $n > 2p$, it results

that a_h for some $h \in [r]$ cannot include more than two distinct symbols from w, hence $r \geq k+1 > i+1$ and there are at least $k > i$ distinct indices h such that $1 < h < r+1$ and $|\alpha_h| > p$, which yields a contradiction with condition iv) from the pumping lemma.

It results that $L \notin tlDR\text{-}\mathcal{L}(i)$.

□

Corollary 4.4 *$tlDR\text{-}\mathcal{L}(k) \setminus tlDR\text{-}\mathcal{L}(i) \neq \emptyset$, for any $i, k \in Nat_0$, with $i < k$.*

Proof Let $V = \{b_1, \ldots, b_{2k+1}\}$ be an alphabet and $l \notin V$ be a distinct symbol. From Proposition 4.1, we have that there exists a language L over $V \cup \{l\}$ such that $L_{lb_1 \ldots b_{2k+1}} \subseteq L \subseteq L^{total}_{lb_1 \ldots b_{2k+1}}$ and $L \in tlDR\text{-}\mathcal{L}(k)$. From $i < k$ and from Proposition 4.3, we have that $L \notin tlDR\text{-}\mathcal{L}(i)$. Thus $L \in tlDR\text{-}\mathcal{L}(k) \setminus tlDR\text{-}\mathcal{L}(i)$.

□

Proposition 4.5 *$tlDR\text{-}\mathcal{L}(i) \setminus tlDR\text{-}\mathcal{L}(k) \neq \emptyset$, for any $i, k \in Nat$, with $i < k$.*

Proof Let $V = \{b_1, \ldots, b_{2i+3}\}$ be an alphabet and $l \notin V$ be a distinct symbol. One can prove in a similar way as we did for Proposition 4.1, that there exists a language L over $V \cup \{l\}$ such that $\{l(b_1)^m \ldots (b_{2i})^m (b_{2i+1}b_{2i+2}b_{2i+3})^m \mid m \in Nat_0\} \subseteq L \subseteq L^{total}_{lb_1 \ldots b_{2i+3}}$ and $L_1 \in tlDR\text{-}\mathcal{L}(i)$. A similar kind of D-trivial left dependency grammar should be defined and only the left D-trivial DR-tree corresponding to a sentence $l(b_1)^m \ldots (b_{2i})^m (b_{2i+1}b_{2i+2}b_{2i+3})^m$ with $m \in Nat_0$, must have a slightly different form, but still a maximal number of i gaps.

Suppose to a contradiction that $L \in tlDR\text{-}\mathcal{L}(k)$. It means that there exists a D-trivial left dependency grammar G such that $L = DR\text{-}L(G,k)$. Let $m \in Nat$ be a natural number. Since $w_m = l(b_1)^m \ldots (b_{2i})^m (b_{2i+1}b_{2i+2}b_{2i+3})^m \in L$, there exists a D-trivial left DR-tree Tr_m such that $s(Tr_m) = w_m$. We can suppose - without any loss of generality - that Tr_m is a left bush[1].

We disregard the manner in which leaves are distributed in Tr_m and keeping the spine of Tr_m we build, bottom-up, a left bush Tr'_m in the following way:

[1] Tr_m may include only a bounded "path" prefix, since otherwise we would be able to "pump" in w_m only a part of the symbols $b_1, \ldots, b_{2i+3}$, keeping the same number of gaps for the DR-tree, hence either $L \neq DR\text{-}L(G,k)$ or $L \not\subseteq L^{total}_{lb_1 \ldots b_{2i+3}}$. In this case, the rest of proof would be done for a natural number m greater than the longest "path" prefix in such a D-trivial left Dr-tree Tr_m. Still, the final conclusion would hold.

- We consider all nodes on the spine of Tr_m, bottom-up, from $(1,3)$ to Rt (the leaf $(1,1)$ will remain in the same position).
- For any node of the spine, we arrange its right daughter (which is obviously a leaf) annotated with a symbol b_x in a sequence bounded to the left by the leaf $((x-1)m+2,1)$ and to the right by the leaf $(xm+1,1)$.
- The leaves will be arranged consecutively on a first-met-first-arranged base, from left to right for leaves annotated with a symbol b_{2z-1} with $z \in [i+2]$ and from right to left for leaves annotated with a symbol b_{2z} with $z \in [i+1]$.

After the left bush Tr'_m is completed, we observe that:

- $s(Tr'_m) = l(b_1)^m \dots (b_{2i+3})^m$.
- $Tr'_m \in T(G)$.
- Moreover, $Tr'_m \in T(G, i+1)$ (it is easy to see that at any moment we have at most $i+2$ islands of natural numbers in the coverage of any node in Tr'_m, thus Tr'_m cannot have more than $i+1$ gaps).

Since $i < k$, it results that also $Tr'_m \in T(G,k)$. Therefore, $l(b_1)^m \dots (b_{2i+3})^m \in$ *DR-L(G,k)*. We obtain that $\{l(b_1)^m \dots (b_{2i+3})^m \mid m \in Nat_0\} \subseteq L \subseteq L^{total}_{lb_1 \dots b_{2i+3}}$. But, using this last statement, the fact that $i < j$ and Proposition 4.3, we have that $L \notin$ *tlDR-*$\mathcal{L}(i)$, which is a contradiction with the initial assumption.

It follows that $L \in$ *tlDR-*$\mathcal{L}(i) \setminus$ *tlDR-*$\mathcal{L}(k)$.

□

Now, we ca state the main result of this paper.

Theorem 4.6 *The classes of languages, generated by D-trivial left dependency grammars with a bounded number of gaps, form an infinite sequence:*

$$tlDR\text{-}\mathcal{L}(1), tlDR\text{-}\mathcal{L}(2), \dots, tlDR\text{-}\mathcal{L}(n), \dots$$

such that any two different classes in this sequence cannot be compared.

Proof It results from Corollary 4.4 and Proposition 4.5.

□

5 Other properties

We denote with REG, CF and CS the classes of regular, context-free, and respectively, context-sensitive languages.

Proposition 5.1 *The following property holds: DR-$\mathcal{L}(0) = CF$.*

Proof If $G = (N, T, S, P)$ is a dependency grammar then we can define a context- free grammar $G' = (N, T, S, P')$ with $P' = \{(u, v) \mid (u, v, z) \in P\} \cup \{(u, v) \mid (u, v) \in P\}$. It results $L(G') = DR\text{-}L(G,0)$. If G is a context-free grammar (we can consider G in Chomsky Normal Form), then we can define a dependency grammar $G' = (N, T, S, P')$ with $P' = \{(A, BC, L) \mid (A, BC) \in P\} \cup \{(A, a) \mid (A, a) \in P\}$[2]. It results $DR\text{-}L(G', 0) = L(G)$.

□

Let $V = \{a_1, \ldots, a_n\}$ be an alphabet. We define the Parikh mapping $\Phi_V : V^* \to Nat_0^n$ by $\Phi_V(x) = (|x|_{a_1}, \ldots, |x|_{a_2})$, for any $x \in V^*$. We extend the mapping in a natural way to languages. Two languages $L_1, L_2 \in V^*$ such that $\Phi_V(L_1) = \Phi_V(L_2)$ are said to be *letter equivalent*. Consider the operations of componentwise addition and multiplication by a constant over the set of natural vectors of a given dimension. A subset M of Nat_0^n is said to be *linear* if there exist the vectors $v_0, v_1, \ldots, v_m$, $m \geq 0$, such that $M = \{v_0 + \sum_{i=1}^{m} p_i v_i \mid p_i \in Nat_0, 1 \geq i \geq m\}$. A finite union of linear sets is called a *semilinear* set. A language $L \subseteq V$ is called *semilinear* if $\Phi_V(L)$ is a semilinear set.

Proposition 5.2 *Let $L \in DR\text{-}L(i)$, for some $i \in Nat$. Then L is semilinear.*

Proof Let G be a dependency grammar such that $L = DR\text{-}L(G,i)$. Denote $L_0 = DR\text{-}L(G, 0)$. From Proposition 2.5, we know that any DR-tree $Tr \in T(G, i)$ is DR-equivalent with a projective DR-tree $Tr_0 \in T(G, 0)$. Then $\Phi_V(s(Tr)) = \Phi_V(s(Tr_0))$. Since $L_0 \subseteq L$ (Claim 1), it results that $\Phi_V(L) = \Phi_V(L_0)$, i.e. L is letter equivalent to L_0. But L_0 is a context-free language (Proposition 5.1) and from the Parikh Theorem it is semilinear. It results that also L is semilinear.

□

[2] G' can be easily brought to the form of a left dependency grammar

Proposition 5.3 *The following properties hold.*

i) $tlDR\text{-}\mathcal{L}(0) = REG$.

ii) $REG \subset tlDR\text{-}\mathcal{L}(i)$, *for any* $i \in Nat$ *(all inclusions are strict).*

iii) $tlDR\text{-}\mathcal{L}(i) \subset CS$, *for any* $i \in Nat_0$ *(all inclusions are strict).*

Proof

i) If G is a regular grammar, then we can define a D-trivial left dependency grammar $G' = (N, T, S, P')$ with $P' = \{(A, aB, L) \mid (A, aB) \in P\} \cup \{(A, a) \mid (A, a) \in P\}$. It results $DR\text{-}L(G', 0) = L(G)$. Let $G = (N, T, S, P)$ be a D-trivial left dependency grammar and $N = N_1 \cup N_2 \cup N_3$ be the partition of the set of nonterminals defined in Proposition 3.3. We can define a non-deterministic finite automaton $M = (Q, T, S, F, \delta)$ such that $Q = N \cup \{B^A \mid A \in N_2, B \in N_3\} \cup \{qF\} \cup \{p_b \mid (A, ab, L) \in P, A \in N_1\} \cup \{r^B_{A,b} \mid (A, ab, L) \in P, B \in N_2, A \in N_3\}$, $F = \{q_f\}$ and the transition function δ is defined by:

$$\delta(q,a) = \begin{cases} \{qF \mid (q,a) \in P\} \cup \{q' \mid (q, aq', L) \in P\} \cup & \\ \cup \{p_b \mid (q, ab, L) \in P\} & \text{if } q = S, \\ \{q' \mid (q, aq', L) \in P\} \cup \{p_b \mid (q, ab, L) \in P\} & \text{if } q \in N_1 \backslash \{S\}, \\ \{p_b, r^q_{A,b} \mid (q, ab, L) \in P\} & \text{if } q \in N_2, \\ \{qF\} & \text{if } q = p_a, \\ \{B^A\} & \text{if } q = r^A_{B,a}, \\ \{C^A \mid (C, Ba, L) \in P, C \in N_3\} \cup & \\ \{qF \mid (A, Ba, L) \in P\} & \text{if } q = B^A, \\ \emptyset & \text{otherwise.} \end{cases}$$

We have *L(M)=DR-L(G,0).*

ii) Like above, if G is a regular grammar, we can define a D-trivial left dependency grammar $G' = (N, T, S, P')$ with $P' = \{(A, aB, L) \mid (A, aB) \in P\} \cup \{(A, a) \mid (A, a) \in P\}$ and we have $DR\text{-}L(G', 0) = L(G)$. But G creates only projective DR-trees ($T(G', i) = T(G', 0)$, for any $i \in Nat_0$), hence $DR\text{-}L(G', i) = DR\text{-}L(G', 0) = L(G)$, for any $i \in Nat_0$. If $i \in Nat$, from the item i) of this proposition and from Corollary 4.4, we obtain the strictness of the inclusion.

iii) Let $L \in DR\text{-}\mathcal{L}(i)$, $i \in Nat_0$ be a language, $G = (N, T, S, P)$ be a D-trivial left dependency grammar such that $L = DR\text{-}L(G,i)$ and $N = N_1 \cup N_2 \cup N_3$ is the partition of N defined in Proposition 3.3. We can define a linear bounded automaton $M = (Q, T, \Sigma, \$, \#, B, q0, F, \delta)$ with a work space of $2n + 3$ cells where n is the length of the input word. Consider the tape alphabet $\Sigma = T \cup \{B, \$, \#, \&\}$, where B is a special symbol marking the cells that are not visited yet, \$ and # are the left (respectively, right) side markers and & is a symbol which will mark the middle of the tape (none of these four tape symbols is in T). $q0$ is the initial state of the automaton M, while the set of final states is defined by $F = \{qF\}$. The set of states Q and the transition function δ are defined according to the method described in the following.

We define an instant configuration of M as a triple (q, α, i), where $q \in Q$, $\alpha \in \Sigma^+$, $|\alpha| = 2n + 3$ and $1 \leq i \leq 2n + 3$. We will use instant configurations to describe the behavior of M.

The initial configuration of M is:

$$(q_0, \$wB^{n+1}\#, 2),$$

with $w \in T^+$, $|w| = n$. In the first step, the r/w-head goes to the middle of the tape, marks the middle with & and goes one cell to the right:

$$(q_S, \$w\&B^n\#, n + 3),$$

where S is the start symbol of the grammar G. The r/w-head starts to write on the tape to the right symbols from T, according to productions from P:

$$(q_A, \$w\&xB^{n-k}\#, n + k + 3) \vdash (q_B, \$w\&xaB^{n-k-1}\#, n + k + 4),$$

if $(A, aB, L) \in P$. To some state q_A, with $A \in N_2 \cup N_3$, M decides non-deterministically to stop this step and to go to the end of the work space.

$$(p_A, \$w\&xB^{n-k}\#, 2n + 2).$$

The r/w-head starts writing to the left, again symbols from T, according to productions from P:

$$(p_A, \$w\&xB^{n-k-l}y\#, 2n + 2 - l) \vdash (p_B, \$w\&xB^{n-k-l-1}ay\#, 2n + 1 - l),$$

if $(A, Ba, L) \in P$, $k + l < n - 2$. When $k + l = n - 2$, two terminals a and b from a production $A \rightarrow_L ab$, with $a, b \in T$ are written on the tape, the $n + k + 2$-th symbol is marked with an overline and the r/w-head goes to the beginning of the work space:

$$(r, \$w\&u\overline{a}y\#, 2),$$

where $uay \in T^+$, $a \in T$. From this point on, a prefix of the input string w is compared letter-by-letter to ua. They should be identical. If this marking procedure succeeds, we reach the configuration:

$$(s, \$\overline{ua}z\&\overline{ua}y\#, n + k + 3).$$

Now, we should find a match between symbols from z and symbols from y, but these have not to be in the same order. We should memorize continuously (under the name of the state) an indicator for the number of gaps induced by this matching. The matching will be carried on in the following way: take the next unmarked symbol from the second half of the work space, go with it to the left, find (and mark) an equal unmarked symbol in the first half of the work space.

We may have three cases:

- If the new marked symbol is not near (to the left or to the right of) an already marked symbol, then the indicator for the number of gaps will increase with 1 (a new gap is created). Obviously, the indicator cannot be ever greater than i.

  $$(s_a^j, \$\alpha bac\beta, m) \vdash (t^{j+1}, \$\alpha b\overline{a}c\beta, m + 1),$$

  where $m = |\$\alpha ba|$, $a, b \in T$, $c \in T \cup \{\&\}$ and $j < i$.
- If the new marked symbol is near an already marked symbol, but not to the left and to the right in the same time, then the indicator for the number of gaps will remain unchanged.

  $$(s_a^j, \$\alpha\overline{b}ac\beta\&\gamma\#, m) \vdash (t^j, \$\alpha\overline{ba}c\beta, m + 1),$$

  where $m = |\$\alpha ba|$, $a, b \in T$, $c \in T \cup \{\&\}$, or

  $$(s_a^j, \$\alpha ba\overline{c}\beta, m) \vdash (t^j, \$\alpha b\overline{ac}\beta, m + 1),$$

  where $m = |\$\alpha ba|$, $a, b, c \in T$.

– If the new marked symbol is near an already marked symbol, both to the left and to the right, then the indicator for the number of gaps will decrease with 1 (a gap was filled-in).

$$(s_a^j, \$\alpha\overline{b}a\overline{c}\beta, m) \vdash (t^{j-1}, \$\alpha\overline{abc}\beta, m+1),$$

where $m = |\$\alpha ba|$, $a, b, c \in T$.

If this marking procedure succeeds, then the final transition, which leads the automaton in an accepting configuration, is described by:

$$(s^j, \$\overline{w}\&\overline{xy}\#, 2n+3) \vdash (qF, \$\overline{w}\&\overline{xy}\#, 2n+2).$$

The detailed definitions of Q and δ are left to the reader. We have *L(M)=DR-L(G,i)*. Since all languages in *tlDR-*$\mathcal{L}(i) \subseteq$ *DR-*$\mathcal{L}(i)$ are semilinear (Proposition 5.2), we can easily find a non- semilinear language in $CS \setminus$ tlDR-$\mathcal{L}(i)$ (like $\{a^{2^n} \mid n \in Nat_0\}$).

□

6 Conclusions

The main aim of this contribution was to discuss the fact that there is a significant difference between the (non)projectivity of DR-trees and respectively D-trees. We stressed on the fact that D-trees can hide some concurrency and word-order freedom phenomena raising in the generation or the parsing of the sentence. D-trivial left dependency grammars and the global DR-restrictions of the word-order freedom were used as the possible most simplest combination of notions useful for this kind od discussion. As outcome an infinite sequence of incomparable classes of semilinear languages bounded between the class of regular and respectively context-sensitive languages was obtained. The massive incomparability achieved trough the global restrictions only is the main novelty of this contribution. This result strengthens the results from [4], where some infinite hierarchies of classes of languages were obtained. Those hierarchies were obtained by using stronger combinations of local and global restrictions applied to free-order dependency grammars.

In the close future, we will study the same types of global word-order restrictions as here, but applied on dependency grammars without any further restrictive condition, like D-triviality. We believe that we will achieve new

sequences of incomparable classes of languages. Moreover, we believe that to this aim we can use the sequence of "witness" languages, which we already used in this paper.

We will study also free-order dependency grammars with several kinds of topological restrictions in order to understand complex word-order and concurrency phenomena occurring in the syntax of natural languages. We believe that the study of free-order dependency grammars can also contribute to the understanding of concurrency phenomena, in general, as well.

Acknowledgements

The authors acknowledge Tomáš Holan for fruitful discussions and a critical reading of the paper.

References

[1] A.Ja. Dikovskij, L.S. Modina : "Dependencies on the other side of the curtain", In: Traitement Automatique des Langues (TAL), 41(1):79-111, 2000

[2] T.Holan, V.Kuboň, M.Plátek : "A Prototype of a Grammar Checker for Czech", In: *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Association for Computational Linguistics, Washington, March 1997, pp.147-154

[3] T. Holan, V.Kuboň, K.Oliva, M.Plátek: "Two Useful Measures of Word Order Complexity", In: *Proceedings of the Coling '98 Workshop "Processing of Dependency-Based Grammars"*, A. Polguere and S. Kahane (eds.), University of Montreal, Montreal, 1998

[4] M.Plátek, T.Holan, V.Kuboň : On Relax-ability of Word-Order by D-grammars., In: *Combinatorics, Computability and Logic*, C.S. Calude and M.J. Deneen (eds.), Springer Verlag, Berlin, 2001, pp. 159-174

[5] P.Sgall, E.Hajičová, J.Panevová: *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*, Dordrecht: Reidel and Prague: Academia, 1986