V. Kůrková et al. (Eds.): ITAT 2014 with selected papers from Znalosti 2014, CEUR Workshop Proceedings Vol. 1214, pp. 68–71
http://ceur-ws.org/Vol-1214, Series ISSN 1613-0073, © 2014 M. Plátek

Analysis by Reduction of D-trees *

Martin Plátek

Charles University in Prague, Faculty of Mathematics and Physics
Malostranské nám. 25, 118 00 Prague, Czech Republic
martin.platek@ufal.mff.cuni.cz

Abstract: The goal of this study is to introduce and observe analysis by reduction of dependency trees. We focus on formalization of certain minimalistic properties of analysis by reduction of dependency trees, and its complexity issues.

1 Introduction

We introduce *analysis by reduction of* D-*trees* (ARDT) in order to give a new formal tool for study of individual dependency trees in tree-banks, namely for the study of their structural complexity. In this way we give at the same time a tool for observations of properties of the whole tree-banks, and their interesting subsets. We work in a way which is close to the formal languages theory. We should be able to study complexity issues of formal D-tree languages.

1.1 Analysis by Reduction Based on D-Trees

Analysis by reduction on D-*trees* is derived from analysis by reduction of natural language sentences (AR), see [3], helps one to study AR, and to identify a sentence dependency structure and the corresponding grammatical categories of an analyzed language. ARDT is based upon a stepwise simplification of a correctly composed D-tree (e.g, correctly composed to a correct Czech sentence). It defines possible sequences of reductions in the D-tree – each step of ARDT consists in some *cuts of* some subtrees from the input D-tree; here, we allow the cuts of to be accompanied by some *shifts* of (a) word(s) (node(s)) to another horizontal position(s) in the D-tree.

Let us stress the basic constraints imposed on reduction steps of surface ARDT:

(i) individual words (word forms), their morphological characteristics and/or their syntactic categories must be preserved in the course of ARDT;

(ii) a (grammatically) correct D-tree must remain correct after its simplification;

(iii) shortening of any reduction would violate the principle of correctness

(iv) a D-tree D which contains a correct D-tree D' as a subtree, where D and D' have the same root, must be further reduced;

(v) specially, an application of the shift operation is limited only to cases when a shift is enforced by the correctness preserving principle (ii), i.e., a reduction consisting of some cuts of only would result in an incorrect word order.

Example 1.
(1) Petr.Sb se.AuxT bojí.Pred o.AuxP otce.Obj ..AuxK
'Peter – REFL – worries – about – father'
'Peter worries about his father.'

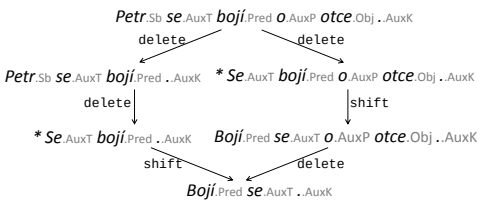

Figure 1: The schema of AR for sentence (1).

The analysis by reduction of sentence (1) can be summarized with the scheme in Fig.1. Our example sentence can be simplified in two ways (for simplicity, we do not make distinction between upper and lower case letters at the beginning of the sentence):
(i) either by deletion of the prepositional group *o otce* 'for father' (according to the correctness constraint on the simplified sentence, the pair of word forms must be deleted in a single step, see the left branch of the scheme);
(ii) or by deleting the subject *Petr* (the right part of the scheme); however, this simplification results in the incorrect word order variant starting with the clitic *se* (such position of a clitic is forbidden in Czech); thus the shift operation is enforced $\rightarrow_{shift}$ *Bojí se o otce.* '(he) worries about his father.'.
As these possible reductions are independent of each other, we can conclude that the words *Petr* and *o otce* 'for father' are independent – in other words, both of them depend on / modify a word remaining in the simplified sentence, i.e., the verb and its clitic *bojí se* '(he) worries'.

Then, the reduction proceeds in a similar way in both branches of AR until the minimal correct simplified sentence $\rightarrow$ *Bojí se.* '(He) worries.' is obtained. This sentence cannot be further reduced.

The order of reductions reflects the dependency relations, which are usually encoded in the form of so called dependency tree, see Figure 2. Informally, the words are 'cut from the bottom of the tree'; i.e., a governing node

*The paper reports on the research supported by the grant of GAČR No. P202/10/1333.

must be preserved in a simplified sentence until all its dependent words are deleted, see [2]. In other words, the above described AR of the sentence (1) makes it possible to see the analysis by reduction of the dependency tree for sentence (1).

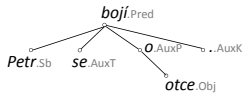

Figure 2: The dependency tree of sentence (1).

2 Formal Apparatus

In the following we introduce and study a formal apparatus which allows an algebraic and complexity characterization (of properties) of analysis by reduction of dependency trees informally outlined in the previous chapter. We work with a finite proper alphabet (vocabulary) Σ_p (modeling individual word forms), an alphabet of categories Σ_c, and a composed basic alphabet $\Gamma \subseteq \Sigma_p \times \Sigma_c$, which models lexico-morphological disambiguations of individual word forms.

To realize a projection from Γ^* to Σ_p^* and Σ_c^*, respectively, we define two homomorphisms, a *proper homomorphism* $h_p : \Gamma \to \Sigma_p$ and a *categorial homomorphism* $h_c : \Gamma \to \Sigma_c$ in the obvious way: $h_p([a,b]) = a$, and $h_c([a,b]) = b$ for each $[a,b] \in \Gamma$. Proper inclusions are denoted by $\subset$.

Example 2. *Proper alphabet :*
Σ_p^e =*{ Petr, se, bojí , o, otce, . }*
Categorial alphabet:
Σ_c^e = *{ Sb, AuxT, Pred, AuxP, Obj, AuxK }*
Basic alphabet :
$\Gamma^e = \{b_1 = [Petr, Sb], b_2 = [se, AuxT], b_3 = [boj, Pred], b_4 = [o, AuxP], b_5 = [otce, Obj], b_6 = [., AuxK]\}$

In this paragraph, we introduce so-called D-structures (Delete or Dependency structures). A D-structure captures syntactic units (words and their categories used in an corresponding sentence) as nodes of a graph and their mutual syntactic relations as edges; moreover, word order is represented by means of total ordering of the nodes.

A D-structure on Σ_p, Σ_c, and Γ is a tuple $D = (V,E,ord(V))$, where the pair (V,E) is a directed acyclic graph, V is its finite set of nodes, $E \subset V \times V$ is its finite set of edges. A *node* $u \in V$ is a tuple $u = [i,a]$, where $a \in \Gamma$ is a symbol (word) with a category assigned to the node, and i is a natural number which serves for an unambigous identification of the node u. Finally, $ord(V)$ is a total ordering of V, usually described as an ordered list of members from V.

Edges are interpreted as representations of syntactic relations between respective lexical units, and $ord(V)$ serves for a representation of the word order in the modelled sentence. Let $ord(V) = ([i_1,a_1],\cdots,[i_n,a_n])$, we say that $w = a_1 \cdots a_n$ is the string (sentence) of D and write $St(D) = w$.

Let $D = (V,E,ord(V))$ be a D-structure, $card(V) = n$, and $ord(V) = ([1,a_1],[2,a_2],\cdots,[n,a_n])$ for some $a_1,\cdots,a_n$. We say that D is a normalized D-structure. Let $D = (V,E,ord(V))$ be a D-structure, $D_1 = (V_1,E_1,ord(V_1))$ a normalized D-structure, (V,E) and (V_1,E_1) are isomorphic, and $St(D) = St(D_1)$. We say that D_1 is a normalization of D.

We say that two D-structures are equivalent if they have equal normalizations.

In the following we usually do not distinquish between equivalent D-structures. Note that to any D-structure is its normalization unambiguously determined.

Further we will work mostly with special D-structures called D-trees. We say that a D-structure $D = (V,E,ord(V))$ on Σ_p, Σ_c, and Γ is a D-tree on Σ_p, Σ_c, and Γ if (V,E) is a rooted tree (i.e., all maximal paths in (V,E) end in its single root).

Example 3. D-*strom representing Figure 2:*
$Tr_1 = (\{[1,b_1],[2,b_2],[3,b_3],[4,b_4],[5,b_5],[6,b_6]\},$
$\{([1,b_1],[3,b_3]),([2,b_2],[3,b_3]),([4,b_4],[3,b_3]),$
$([5,b_5],[4,b_4]),([6,b_6],[3,b_3])\},$
$([1,b_1],[2,b_2],[3,b_3],[4,b_4],[5,b_5],[6,b_6]))$

Let $D = (V,E,ord(V))$, $D_1 = (V_1,E_1,ord(V_1))$, where (V_1,E_1) is a subtree of (V,E) which contains the root of D, and $ord(V_1)$ is a permutation of a subsequence of $ord(V)$. Then we write $D_1 \subset D$.

Example 4. D-*strom representing a subtree of* Tr_1*:*
$Tr_2 = (\{[1,b_1],[2,b_2],[3,b_3],[6,b_6]\},$
$\{([1,b_1],[3,b_3]),([2,b_2],[3,b_3]),([6,b_6],[3,b_3])\},$
$([1,b_1],[2,b_2],[3,b_3],[6,b_6]))$
We can see that $Tr_2 \subset Tr_1$.

We say that a set T of D-trees on Σ_p, Σ_c, and Γ is a D-language on Σ_p, Σ_c, and Γ. We write $T \in \mathsf{D}(\Sigma_p,\Sigma_c,\Gamma)$. We say that $St(T)$ is the *string language of* T, $h_p(St(T))$ is the proper language of T, and $h_c(St(T))$ the categorial language of T.

Now we introduce a basic operation for analysis by reduction on D-trees. It is determined by a node u of a D-tree which is not the root of the D-tree We call this operation an *u-cut)*. Note that any node u of a D-tree D, where u is not the root of D, unambiguously determines a partition of D into two subtrees:

1) lower subtree $T_L(u,D)$ of D by u, i.e. such a maximal subtree of D containing only nodes on some path leading to u (including the node u),

2) upper subtree $T_U(u,D)$ of D by u, i.e. such a maximal subtree of D containing the root of D, and all nodes which are not from $T_L(u)$. An u-cut on D transforms D to $T_U(u,D)$. We denote such a u-cut (directly) by $T_U(u,D)$. We can see that $T_U(u,D)$ is again a D-tree .

Now we introduce an operation called *shift*, suitable for an enhancement of analysis by reduction on D-trees.

A shift means a move of a single node of a D-tree to some new place in its ordering, i.e., only the total ordering of the set nodes is changed.

Consider u-cuts and shifts being the only operations allowed on (a set of) D-trees. Based on these operations, we can naturally define a partial order $\succ_T$ on a D-language T. We say that t_1 is *Dt-reduced* to t_2 in T and write $t_1 \succ_T t_2$ iff:

1. t_2 is is obtained from t_1 by a sequence O of u-cuts possibly followed by some shifts (O contains at least one u-cut); $t_1 \stackrel{O}{\rightarrow} t_2$;
2. the application of any proper subsequence O' of O on u would end up with a D-tree outside T.
3. Let $T_U(u,D)$ be an u-cut from O. Any substitution of $T_U(u,D)$ by some $T_U(v,D)$ in O such that $T_U(u,D) \subset T_U(v,D) \subset D$ would end up with a D-tree outside T.

By $\succ_T^+$ we denote transitive, non-reflexive closure of $\succ_T$.

Example 5.
$Tr_3 = (\{[2,b_2],[3,b_3],[6,b_6]\},$
$\{([2,b_2],[3,b_3]),([6,b_6],[3,b_3])\},([3,b_3],[2,b_2],[6,b_6]))$

Let $T_1 = \{Tr_1, Tr_2, Tr_3\}$. *We can see that*

$$Tr_1 \succ_{T_1} Tr_2 \succ_{T_1} Tr_3,$$

and that the Dt-reduction $Tr_2 \succ_{T_1} Tr_3$ *uses a shift.*

The partial order $\succ_T$ naturally defines the set $L^{min}_{\succ_T}$ of minimal D-trees in T:

$$L^{min}_{\succ_T} = \{v \in T \mid \neg\exists u \in T : v \succ_T u\}.$$

Example 6.
$Tr_4 = (\{[2,b_2],[3,b_3],[4,b_4],[5,b_5],[6,b_6]\},$
$\{([2,b_2],[3,b_3]),([4,b_4],[3,b_3]),([5,b_5],[4,b_4]),$
$([6,b_6],[3,b_3])\},$
$([3,b_3],[2,b_2],[4,b_4],[5,b_5],[6,b_6]))$

Let $T_2 = \{Tr_1, Tr_2, Tr_3, Tr_4\}$. *We can see that*

$$Tr_1 \succ_{T_2} Tr_2 \succ_{T_2} Tr_3,\ \ Tr_1 \succ_{T_2} Tr_4 \succ_{T_2} Tr_3,$$

, and that the Dt-reductions $Tr_2 \succ_{T_2} Tr_3$, *and* $Tr_1 \succ_{T_2} Tr_4$ *use a shift.*

Further we can see that $L^{min}_{\succ_{T_2}} = \{Tr_3\}$.

2.1 Analysis by Reduction for a D-Language

Let T be a D-language. We write $DtP(T) = \{\, u \succ_T v | u,v \in T\}$. We say that $DtP(T)$ is the DS-precedence set for T.

We say that $DtP(T)$ is an *analysis by reduction* $AR(T)$ for T if $AR(T) = DtP(T)$, and $T = L^{min}_{\succ_T} \cup \{v; \exists u, v \succ_T u \in AR(T)\}$, and $L^{min}_{\succ_T} \subseteq \{v; \exists u, u \succ_T v \in AR(T)\}$.

We can see that $DtP(T)$ is determined unambiguously by T. Therefore, if T have some analysis by reduction, the analysis by reduction $AR(T)$ is determined unambiguously by T.

Example 7. *We can see that*
$AR(T_2) = \{Tr_1 \succ_{T_2} Tr_2,\ Tr_2 \succ_{T_2} Tr_3,\ Tr_1 \succ_{T_2} Tr_4,$
$Tr_4 \succ_{T_2} Tr_3\}$.

2.2 Lexical Complexity

Let $T \in \mathsf{D}(\Sigma_p, \Sigma_c, \Gamma)$ be a D-language. We suppose in the following that the alphabets (vocabularies) $\Sigma_p, \Sigma_c, \Gamma$ are minimal due to T, i.e., any removing of some symbol of some of this alphabets would cause some removing of some D-tree(s) from T. The syntactic analysis of natural languages is from the point of view of computational linguistic a very difficult task. One of the main reasons of this fact is the enormous size of vocabularies corresponding to $\Sigma_p, \Sigma_c,$, and Γ. E.g. we have access to vocabulary with circa 700 000 items for Czech word forms. Another source of complexity for the syntactic analysis is the lexical ambiguity in the vocabularies corresponding to Γ. E.g., the Czech word *jarní* has 28 different morphological disambiguations. In the following paragraph we focus on such type of structural complexity measures which will witnesses relative structural simplicity of dependency trees for Czech sentences.

2.3 Structural Complexity

With respect to the previous motivation, we focus on the number of cuts and shifts used in individual Dt-reductions, and on the number of nodes deleted in individual Dt-reductions. We use particular abbreviations for languages with restriction on these complexity measures. In particular, prefix Dt- is used to identify the D-languages without any restriction, and Dc- is used for languages with cuts only. Further, the prefix c(k)- is used to indicate that at most k cut-operations are available in one Dt-reduction. We use the syllable de(i)- for languages with at most i deleted nodes in a single Dt-reduction, and s(j)- for languages with at most j shifts in a single Dt-reduction. For each type X of restrictions, we use $\mathscr{D}(\mathsf{X})$ to denote the class of all D-languages with Dt-reductions fullfiling X-restrictions. Further, $\mathscr{D}_n(\mathsf{X})$ denote the classes of D-languages which are determined by D-languages with D-trees which can be reduced by Dt-reductions fullfiling X-restrictions at most n-times.

Let T_2 be the D-language given by Example 6. We can see that $T_2 \in \mathscr{D}_2$(c(1)-s(1)-de(2)-Dt), and that this complexity classification of T_2 is optimal.

Let us consider all purely dependency D-trees from the Prague Tree-Bank ([1]). We believe (by our observations) that this set is a subset from $\mathscr{D}_{100}$(c(1)-s(1)-de(7)-Dt). The most important observation is that the analysis by reduction of pure dependency trees is characterized by one cut in one Dt-reduction. If we should consider also coordinations etc., the situation will be surely more complex.

2.4 Hierarchies

We are able to show that there are infinite hierarchies of formal (in)finite D-languages with analysis by reduction based on the measures $c(i)$, $s(i)$, and $de(i)$.

We plan also introduce and study hierarchies based on the degrees of discontinuity and/or non-projectivity of individual Dt-reductions, and all that to model by restarting automata.

Conclusion and Perspectives

We have introduced analysis by reduction of D-trees in order to formally characterize the basic properties of the dependecy based syntactic analysis of Czech sentences. We will show in the close future that it is a fine tool for the description of structural complexity and ambiguity of natural language (Czech) sentences.

References

[1] Jan Hajič, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský, and Magda Ševčíková-Razímová. *Prague Dependency Treebank 2.0*. Linguistic Data Consortium, Philadelphia, 2006.

[2] Markéta Lopatková, Martin Plátek, and Vladislav Kuboň. Modeling Syntax of Free Word-Order Languages: Dependency Analysis by Reduction. In Matoušek, V. et al., editor, *Proceedings of TSD 2005*, volume 3658 of *LNCS*, pages 140–147. Springer, 2005.

[3] Martin Plátek, Markéta Lopatková, and Dana Pardubská. On Minimalism of Analysis by Reduction by Restarting Automata. In *Accepted for Conference on Formal Grammar 2014*. Springer.